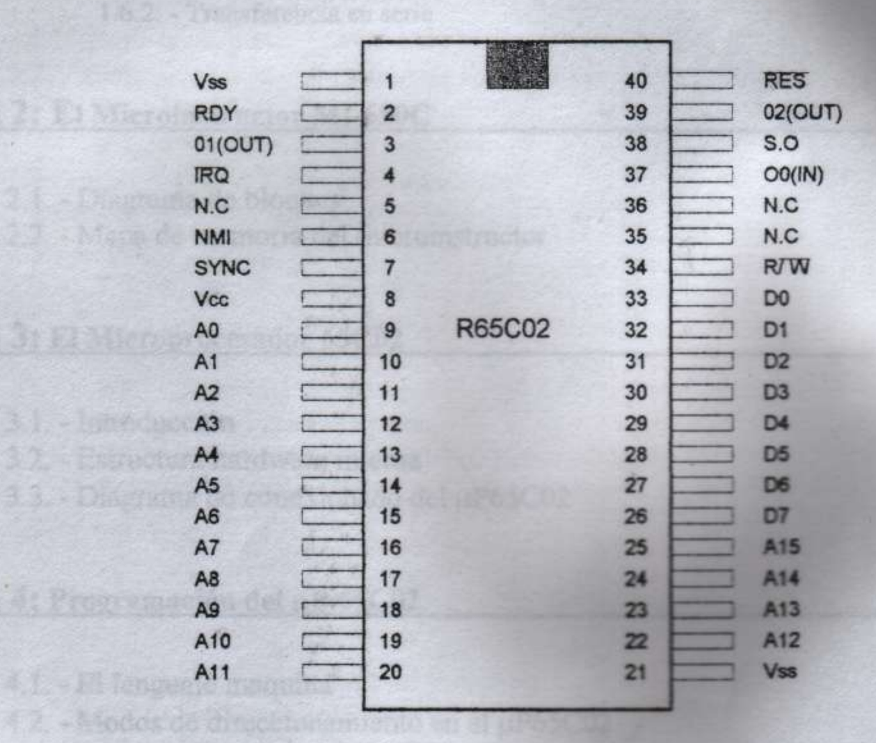
Microprocesadores de 8 Bit



LUCES A4 ØØ

I.F.P." Valle del Sol"
Departamento de Electrónica
Profesores: Elías Aneas Pretel
Joaquín Bernabé Martínez

Indice de Contenido

Tema 1 Estructura de un ordenador

1.1 Estructura y comunicación interna.

La Unidad Central de Proceso ó CPU consta de dos secciones básicas: La Unidad de Control y la Unidad Operativa.

La CPU es la encargada de recibir, interpretar y ejecutar las instrucciones, estando estas últimas grabadas en una memoria interna.

Los Módulos de Entrada/Salida se ocupan de enviar y recibir la información. En la figura 1.1 se pueden apreciar las diferentes partes de que consta un ordenador típico.

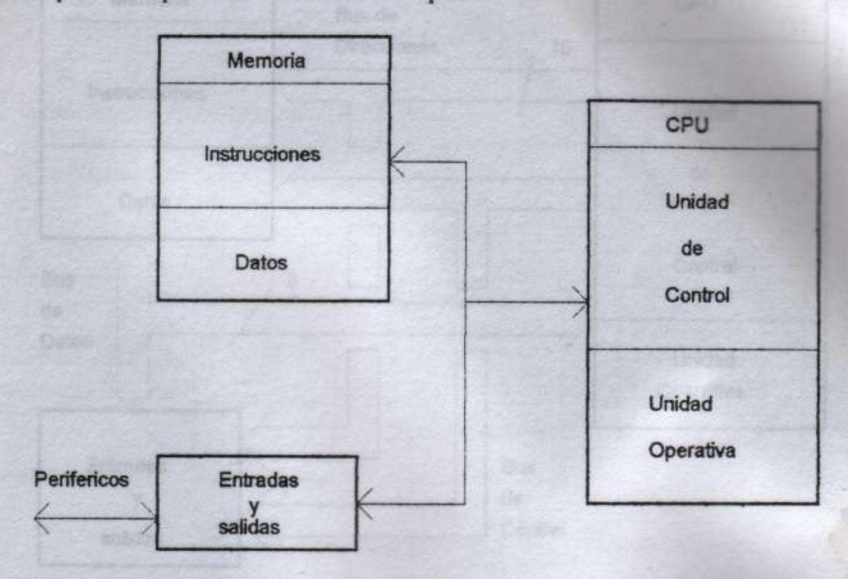


Fig. 1.1 Estructura general de un ordenador

Los distintos bloques que componen un ordenador están comunicados entre sí a través de conjunto de líneas por donde se transporta la información binaria (Buses). La CPU es la encargada de determinar el acceso a la transferencia o intercambio de información de las memorias o de los módulos de I/O (entradas/salidas). Para esta selección usa un colector de líneas digitales denominado Bus de Direcciones. En el Dispositivo que se va a estudiar es común el uso de 16 líneas en el Bus de Direcciones, lo que posibilita una capacidad de selección de $2^{16} = 65536$ (64K) elementos diferentes. Es un Bus de carácter unidireccional pues su contenido lo proporciona siempre la CPU a través de la Unidad de Control. Ver figura 1.2.

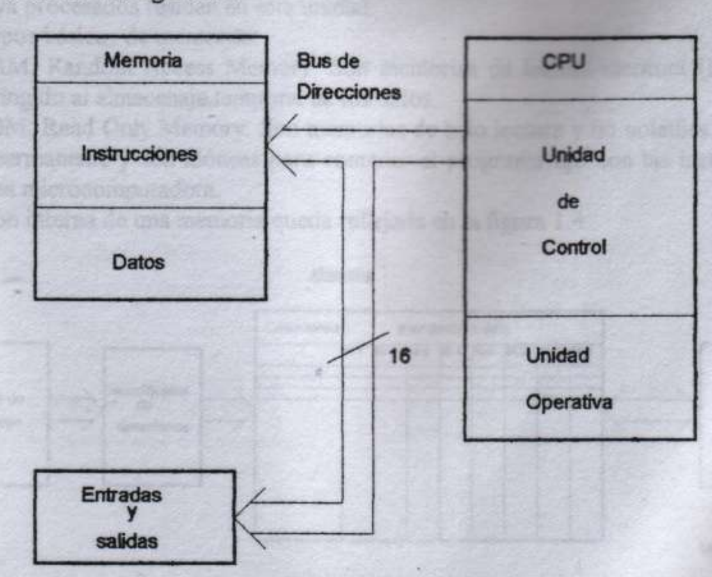


Fig. 1.2 La CPU a través del Bus de Direcciones determina con que elementos realizará la transferencia de información o la operación en curso.

Una vez elegido el elemento que va a trabajar es necesario disponer de una serie de líneas por donde se transfiera la información. A este colector de líneas se le denomina Bus de Datos, en el μP 65C02 esté es de 8 Bits, permitiendo con ello un máximo posible de $2^8 = 256$ instrucciones, por la misma razón las posiciones de memoria ocupadas en la memoria de datos serán de 8 Bits.

El Bus de Datos tiene carácter bidireccional y triestado, pues es compartido por todos los elementos de la computadora. La Unidad de Control establecerá cual será el elemento emisor y cual el receptor de la información.

Existen además otra serie de líneas encargadas de la sincronización de los distintos elementos y por el que circulan una serie de señales auxiliares de gobierno. Entre estas líneas destacan las que transportan impulsos de reloj, las que indican si las operaciones son de escritura o de lectura, de reset o inicialización. Todas estas líneas conforman el denominado Bus de control, como puede apreciarse en la figura 1.3.

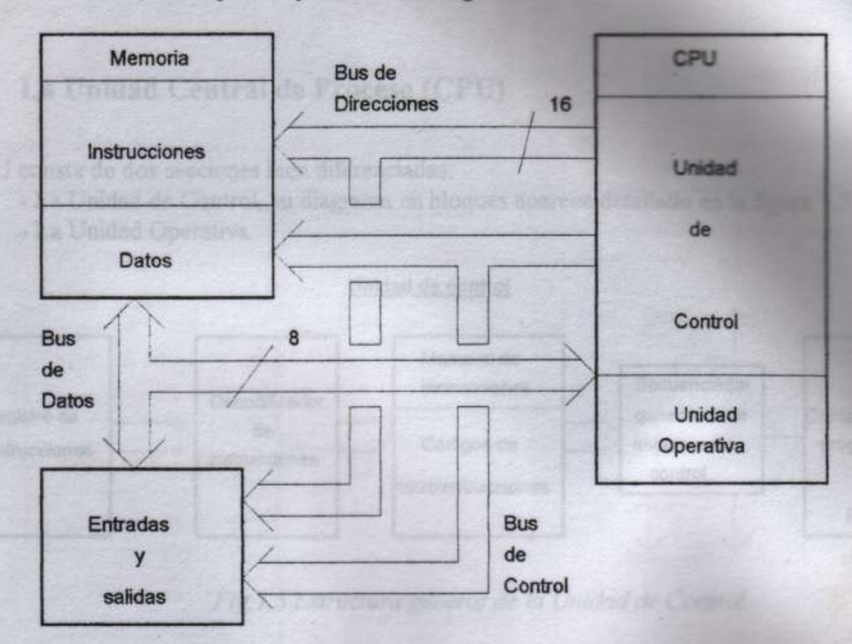


Fig. 1.3 Intercomunicación de los distintos bloques de un ordenador a través de los buses de direcciones, datos y control.

1.2 La Unidad de Memoria

Tanto los códigos binarios de las instrucciones que ha de ejecutar la CPU como los datos o informaciones digitales a procesar o ya procesados residen en esta unidad.

Existen dos tipos básicos de memorias:

- RAM. Random Access Memory. Son memorias de lectura/escritura (R/W), de carácter volátil pues su uso se ve restringido al almacenaje temporal de los datos.

- ROM. Read Only Memory. Son memorias de solo lectura y no volátiles. En estas los datos están contenidos de forma permanente y son idóneas para contener el programa fijo con las instrucciones para el que ha sido diseñado el sistema microcomputadora.

La constitución interna de una memoria queda reflejada en la figura 1.4



Fig.1.4 Estructura interna de una memoria.

Cuando la Unidad de Control envía a través del Bus de Direcciones una que corresponde a la memoria, está es registrada inicialmente para después ser decodificada y obtener la posición de memoria oportuna.

Si la memoria activa es una ROM la transferencia de información se leerá de la memoria, pasando posteriormente su contenido al Bus de datos.

Si la memoria activa es una RAM es necesaria una señal auxiliar de control para determinar si la operación a realizar es de lectura o de escritura. De este modo la información saldrá al Bus de datos (lectura) o bien el contenido del Bus de datos será escrito en la posición ya seleccionada de RAM.

La señal que controla este proceso es la de lectura/escritura ($R/\overline{W}$).

1.3 La Unidad Central de Proceso (CPU)

La CPU consta de dos secciones bien diferenciadas:

- La Unidad de Control, su diagrama en bloques aparece detallado en la figura 1.5
- La Unidad Operativa.



Fig.1.5 Estructura general de la Unidad de Control.

Una vez recibido el código binario de la instrucción a través del Bus de Datos e Instrucciones, esta es registrada y decodificada, lo cual da acceso a la memoria de instrucciones (ROM) donde se encuentran los códigos de las microinstrucciones que componen las distintas etapas parciales que entrañan su ejecución. Las microistrucciones hacen que el secuenciador genere las señales que controlan a los demás elementos del sistema y lleve a cabo la instrucción en curso.

El Contador de Programa (PC) se encarga de enviar por el Bus de Direcciones la posición en donde se encuentra la siguiente instrucción a realizar. Para ello el PC se incrementa en una unidad una vez aceptada la dirección anterior por la memoria, sin embargo, hay instrucciones que le permiten variar su contenido, lo que provoca la rotura del programa y en definitiva da opción a la toma de decisiones.

La Unidad Operativa se comporta como una Unidad Aritmético Lógica (ALU) ampliada. En esta se realizan operaciones lógicas, de rotación de bits, de translación, de transferencia, etc. En la figura 1.6 aparecen detallados los bloques que la componen.

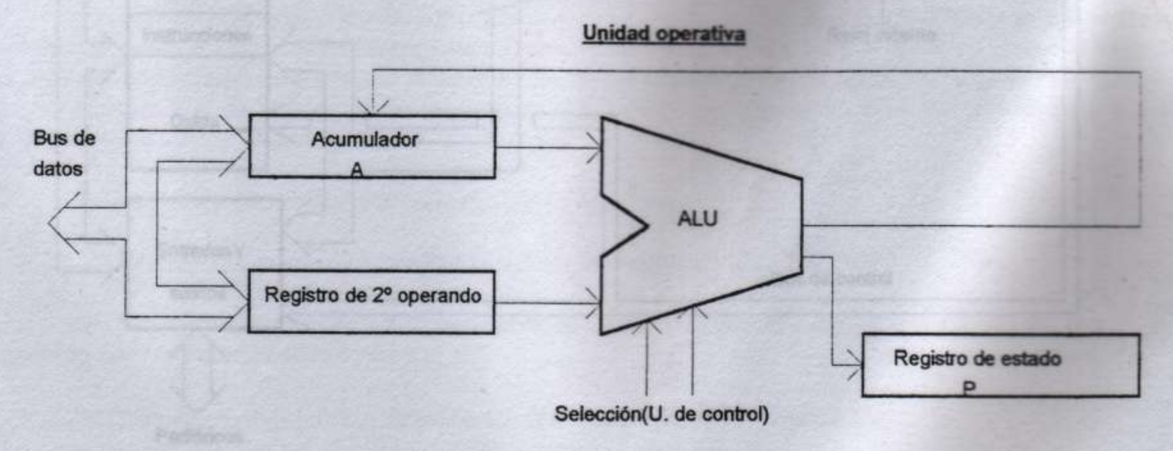


Fig. 1.6 La Unidad Aritmético Lógica.

El Acumulador (A) es un registro especial de 8 bits, encargado de enviar uno de los operandos a la ALU y en donde posteriormente queda depositado el resultado una vez que la operación ha sido efectuada.

La ALU recibe uno de los operandos desde el Acumulador y el otro desde un registro auxiliar(Registro de 2° operando). La operación a realizar es seleccionada mediante las líneas que parten del Secuenciador. El Registro de Estado (P) almacena los sucesos más importantes que suceden en la ALU, actuando como *flags* o señalizadores. Un bit del registro P indica si ha habido acarreo, si el signo del resultado es positivo o negativo, si hay desbordamiento, etc.

Esta forma de trabajar permite que las instrucciones sólo hagan referencia a una dirección de memoria, puesto que la ALU tiene fijada la entrada de un operando y la colocación del resultado en el Acumulador.

La dirección a especificar será aquella que contenga el segundo operando, que habrá que trasladar hasta el registro auxiliar que alimenta a la otra entrada de la ALU.

1.4 Diagrama en bloques de un sistema basado en microprocesador

En la figura 1.7 aparecen los bloques detallados de que consta un sistema con microprocesador.

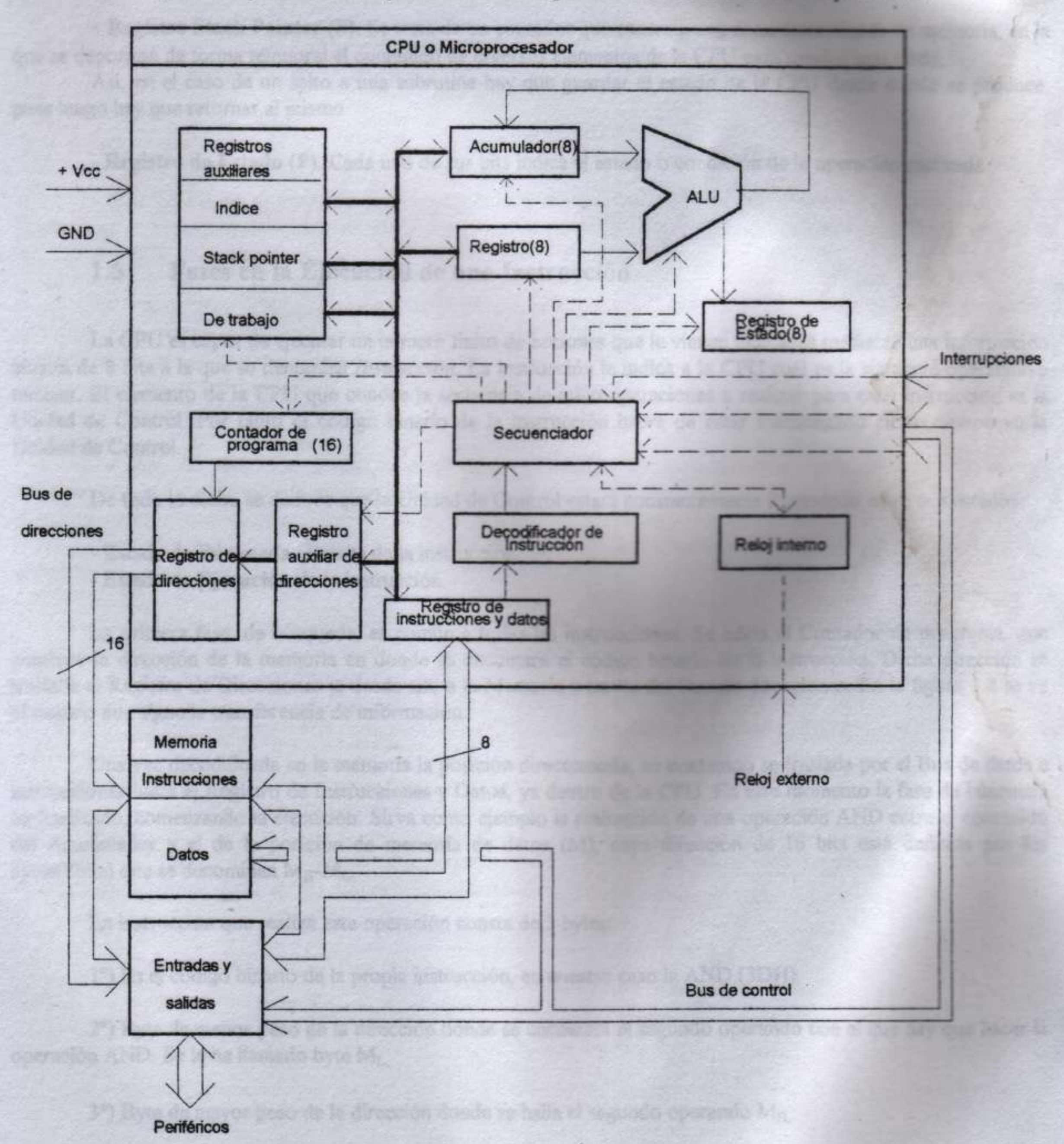


Fig.1.7 Diagrama general de un sistema basado en microprocesador

Todo el sistema funciona de forma adecuada debido a la interrelación de los distintos dispositivos a través de los Buses de Datos, Direcciones y Control.

En el interior del bloque de la CPU están incluidos todos los elementos que conforman la Unidad de Control y la Unidad Operativa. Existen asimismo algunos registros auxiliares que son añadidos de forma arbitraria por los fabricantes de los microprocesadores para facilitar las aplicaciones que ellos recomiendan.

Cabe destacar entre los registros auxiliares:

- **Registro Índice.** Su contenido se emplea para localizar ciertas posiciones de memoria.

- **Registros de Trabajo.** Manipulan datos de interés en el programa y facilitan su búsqueda. El fabricante decide su inclusión.

- **Registro Stack Pointer (S).** Se trata de un contador que controla una zona determinada de memoria, en la que se depositan de forma temporal el contenido de diversos elementos de la CPU para usarlos más tarde.

Así, en el caso de un salto a una subrutina hay que guardar el estado de la CPU desde donde se produce, pues luego hay que retornar al mismo.

- **Registro de Estado (P).** Cada uno de sus bits indica el estado o condición de la operación realizada.

1.5 Fases en la Ejecución de una Instrucción

La CPU es capaz de ejecutar un número finito de acciones que le vienen indicadas mediante una información binaria de 8 bits a la que se denomina *Instrucción.* La instrucción le indica a la CPU cual es la siguiente operación a realizar. El elemento de la CPU que conoce la secuencia de microoperaciones a realizar para cada instrucción es la Unidad de Control. Por tanto el código binario de la instrucción habrá de estar memorizado cierto tiempo en la Unidad de Control.

De todo lo dicho se deduce que la Unidad de Control estará constantemente fluctuando entre dos estados:

- Estado de **Búsqueda (Fetch)** de la instrucción.
- Estado de **Ejecución** de la instrucción.

La primera fase, de búsqueda, es común a todas las instrucciones. Se inicia el Contador de programa, que contiene la dirección de la memoria en donde se encuentra el código binario de la instrucción. Dicha dirección se traslada al Registro de Direcciones y, desde allí, a la Memoria a través del Bus de direcciones. En la figura 1.8 se ve el camino que sigue la transferencia de información.

Una vez decodificada en la memoria la posición direccionada, su contenido se traslada por el Bus de datos e instrucciones hasta el Registro de Instrucciones y Datos, ya dentro de la CPU. En este momento la fase de búsqueda ha finalizado, comenzando la ejecución. Sirva como ejemplo la realización de una operación AND entre el contenido del Acumulador y el de la posición de memoria de datos (M), cuya dirección de 16 bits está definida por los bytes(8bits) que se denominan M_H-M_L.

La instrucción que realiza esta operación consta de 3 bytes:

1°) Es el código binario de la propia instrucción, en nuestro caso la AND (3DH).

2°) Byte de menos peso de la dirección donde se encuentra el segundo operando con el que hay que hacer la operación AND. Se le ha llamado byte M_L.

3°) Byte de mayor peso de la dirección donde se halla el segundo operando M_H.

La memoria de programa contendrá estos bytes con la disposición de la figura 1.8

MAPA DE MEMORIA

Memoria de programa		
Dirección	**Contenido**	
n	Código operación AND	
n+1	ML	← 8 bits de menos peso de la dirección M
n+2	MH	← 8 bits de más peso de la dirección M
Memoria de datos		
MH - ML	2º operando	

Fig. 1.8 Forma de contener en la memoria de programa los tres bytes que componen la instrucción And entre el valor binario del Acumulador y el que contiene la posición de memoria, cuya dirección viene especificada por los bytes M_L y M_H.

La fase de búsqueda de la instrucción se inicia conteniendo el Contador de Programa (PC) la dirección "n", en donde se encuentra el código de la instrucción AND. La dirección "n" en la memoria se decodifica y su contenido sale por el Bus de datos e instrucciones (figura 1.10).

Esta fase termina cuando el código de operación, conteniendo en la posición "n" se deposita en el Registro de Datos e Instrucciones de la CPU.

La fase de ejecución comienza trasladando el código de la instrucción hasta el Decodificador de Instrucciones, quien seleccionará las microinstrucciones que la implementan mediante la generación de señales desde el secuenciador. Dichas señales preparan a la ALU para realizar la operación lógica AND entre el contenido existente en el Acumulador y el de la posición de memoria de datos, cuya dirección viene especificada en los dos bytes siguientes de la memoria de programa.

Preparada la ALU, la siguiente microinstrucción sacará la dirección n + 1 del Contador de Programa, con lo que se localizan los 8 bits de menos peso, M_L, de la dirección M. Estos 8 bits se cargan, temporalmente, en la parte baja del Registro Auxiliar de Direcciones. Después, el Contador de Programa sacará por el Bus de direcciones su nuevo contenido, n + 2, cuya decodificación en la memoria permitirá obtener los 8 bits de más peso, M_H, de la dirección M. Al colocarse estos bits en la parte alta del Registro auxiliar de direcciones, este último queda completo, (figura 1.11).

A continuación, el Registro auxiliar deposita su contenido (M_L-M_H) en el Bus de direcciones, obteniéndose el segundo operando de la zona de la memoria de datos. Dicho operando se traslada por el Bus de datos hasta la CPU, donde se transfiere por su Bus de datos interno, desde el Registro de Instrucciones y datos de entrada hasta el Registro que alimenta, junto con el Acumulador, a la ALU.

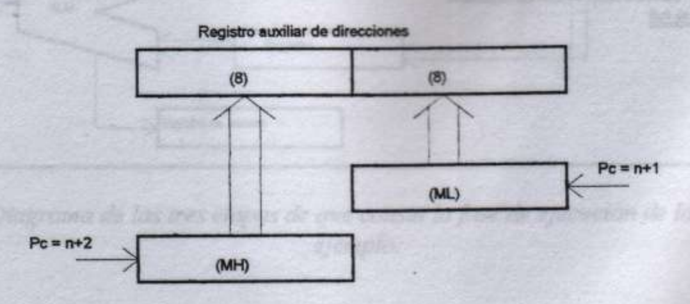


Fig. 1.9 Carga del Registro Auxiliar de Direcciones en dos pasos

Estando disponibles los dos operandos en las entradas de la ALU, ésta realiza la operación AND, bit a bit de los mismos, depositando su resultado en el Acumulador.

En la figura 1.10 se muestra gráficamente la forma de llevarse a cabo la fase de búsqueda de la instrucción AND.

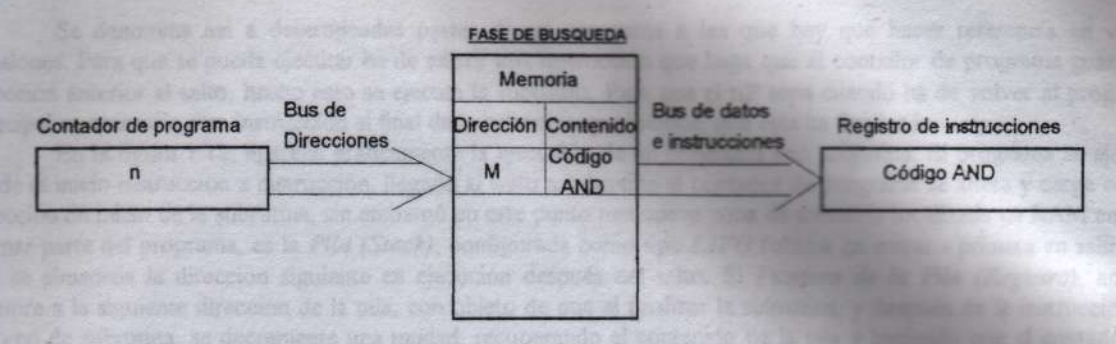


Fig1.10. Fase de búsqueda de la instrucción AND del ejemplo

En la fig1.11 se establece gráficamente la forma de ejecutar la instrucción AND del ejemplo en tres etapas consecutivas.

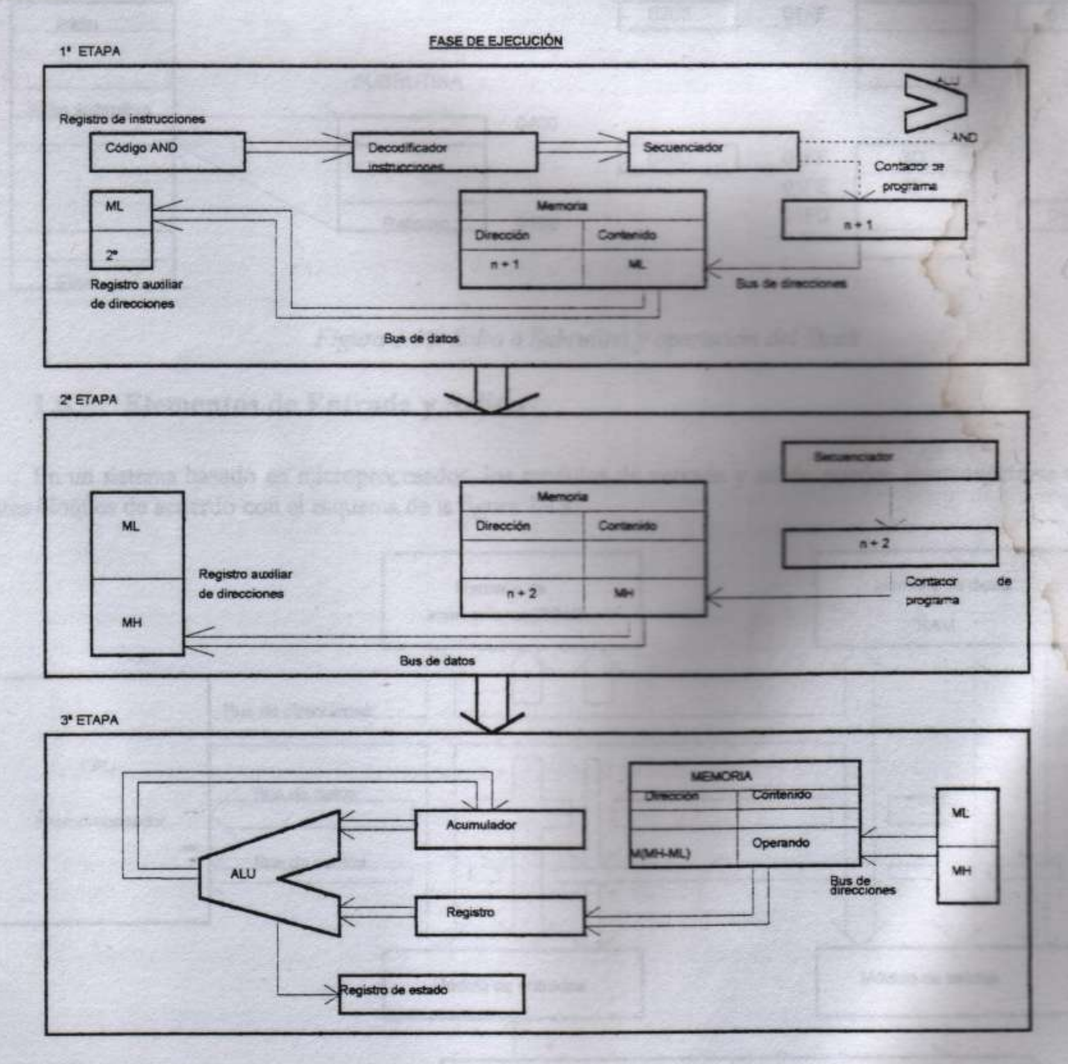


Fig.1.11 Diagrama de las tres etapas de que consta la fase de ejecución de la instrucción AND del ejemplo.

La instrucción comentada llevaba aparejada un dato que, con frecuencia, está ubicado en una posición de la memoria de datos.

Para especificar la instrucción, se ha precisado, además de indicar el código de operación en el primer byte de la misma, añadir dos bytes más que definen la dirección de la memoria donde se encuentra el segundo operando. Recuérdese que nos referimos a un Bus de direcciones compuesto por 16 líneas.

1.5.1 Subrutinas

Se denomina así a determinadas partes de un programa a las que hay que hacer referencia en varias ocasiones. Para que se pueda ejecutar ha de existir una instrucción que haga que el contador de programa guarde la dirección anterior al salto, hecho esto se ejecuta la subrutina. Para que el µP sepa cuando ha de volver al programa principal es necesaria otra instrucción al final de la subrutina que indique, que esta ha finalizado.

En la figura 1.12, aparece gráficamente la ejecución de un programa con subrutina. El programa se ejecuta desde el inicio instrucción a instrucción, llegado al salto a subrutina el contador de programa se altera y carga con la dirección de inicio de la subrutina, sin embargo en este punto una nueva zona de memoria localizada en RAM entra a formar parte del programa, es la ***Pila (Stack)***, configurada como tipo ***LIFO*** (ultima en entrar - primera en salir), en ella se almacena la dirección siguiente en ejecución después del salto. El ***Puntero de la Pila (Registro)***, apunta siempre a la siguiente dirección de la pila, con objeto de que al finalizar la subrutina, y después de la instrucción de retorno de subrutina, se decremente una unidad, recuperando el contenido de la pila y haciendo que el contador de programa se dirija a esta dirección y ejecute la instrucción que sigue al salto.

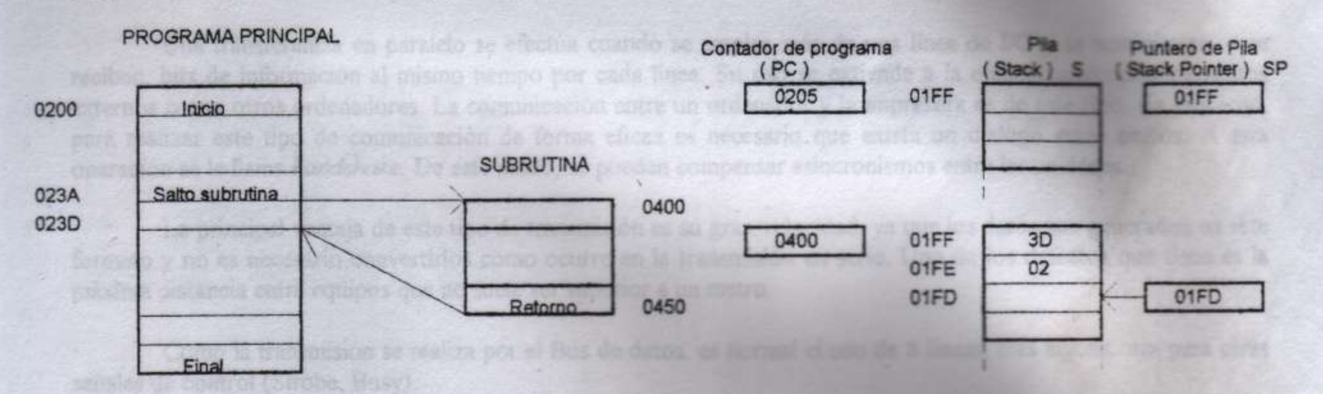


Figura 1.12. Salto a Subrutina y operación del Stack

1.6 Elementos de Entrada y Salida

En un sistema basado en microprocesador, los módulos de entrada y salida pueden interconectarse con los restantes bloques de acuerdo con el esquema de la figura 1.13.

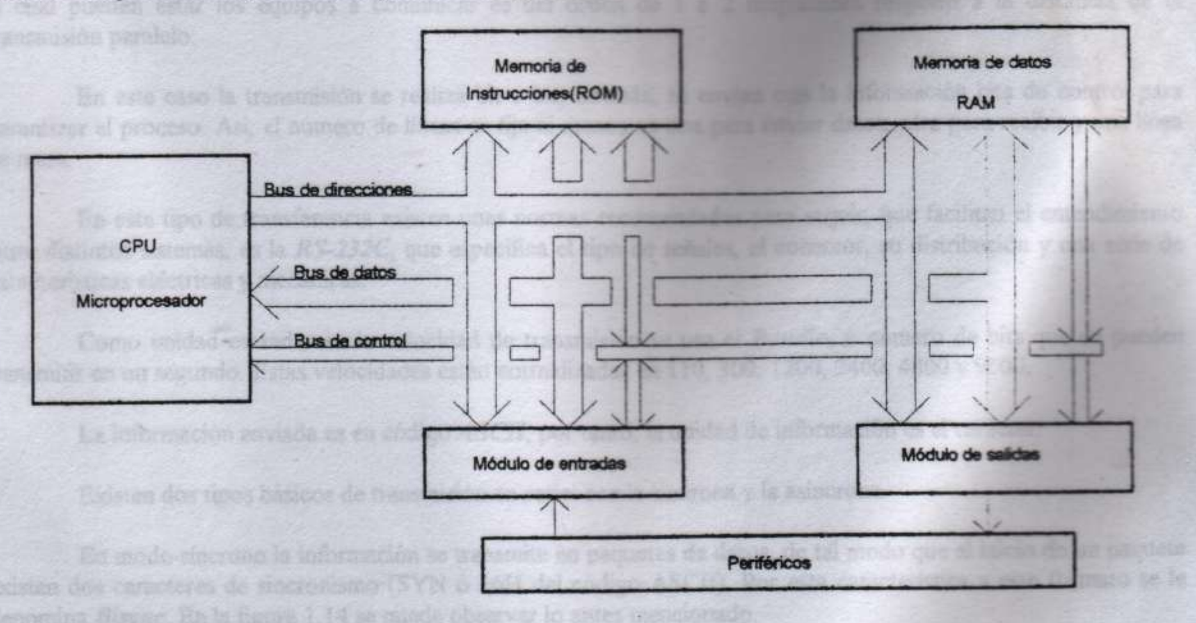


Fig.1.13. Conexión de los módulos de entrada salida a los restantes bloques del sistema.

A estos elementos de transferencia de información con el exterior, se le denomina periféricos. Un ejemplo son el teclado, la pantalla, la impresora, etc.

En la salida de datos el microprocesador se encarga de depositar dicho dato en los puertos de salida, el elemento exterior se encargará de recibirlo del modo preciso.

En la recepción los datos llegan a los puertos de entrada al sistema microcomputador, en este punto los datos son leídos.

Previamente a estas operaciones tan simples, si el periférico es programable, el sistema habrá de configurar al periférico de forma que interprete correctamente la información transferida.

La comunicación entre los elementos de I/O y el exterior puede ser, básicamente, de dos tipos: paralelo y serie.

1.6.1 Transferencia en paralelo

Una transferencia en paralelo se efectúa cuando se emplea más de una línea de I/O y se suministran, o se reciben, bits de información al mismo tiempo por cada línea. Su uso se extiende a la comunicación con elementos externos o con otros ordenadores. La comunicación entre un ordenador y la impresora es de este tipo, sin embargo, para realizar este tipo de comunicación de forma eficaz es necesario que exista un dialogo entre ambos. A esta operación se le llama *handshake*. De este modo, se pueden compensar asincronismos entre las unidades.

La principal ventaja de este tipo de transmisión es su gran velocidad, ya que los datos son generados en este formato y no es necesario convertirlos como ocurre en la transmisión en serie. Uno de los defectos que tiene es la máxima distancia entre equipos que no suele ser superior a un metro.

Como la transmisión se realiza por el Bus de datos, es normal el uso de 8 líneas, más alguna otra para otras señales de control (Strobe, Busy).

1.6.2 Transferencia en Serie

Su uso proviene de la necesidad de reducir el número de líneas para transmitir información. La longitud a la cual pueden estar los equipos a comunicar es del orden de 1 a 2 magnitudes respecto a la distancia de la transmisión paralelo.

En este caso la transmisión se realiza bit a bit, además, se envían con la información bits de control para garantizar el proceso. Así, el número de líneas se fija al menos en una para enviar datos, otra para recibir y una línea de masa.

En este tipo de transferencia existen unas normas recomendadas para seguir, que facilitan el entendimiento entre distintos sistemas, es la ***RS-232C***, que especifica el tipo de señales, el conector, su distribución y una serie de características eléctricas y mecánicas.

Como unidad estándar en la velocidad de transmisión se usa el ***Baudio***, o número de bits que se pueden transmitir en un segundo. Estas velocidades están normalizadas en 110, 300, 1200, 2400, 4800 y 9600.

La información enviada es en código ***ASCII***, por tanto, la unidad de información es el carácter.

Existen dos tipos básicos de transmisión en serie: son la síncrona y la asíncrona.

En modo síncrono la información se transmite en paquetes de datos, de tal modo que al inicio de un paquete existen dos caracteres de sincronismo (SYN ó 16H del código ASCII). Por esta caracteristica a este formato se le denomina ***Bisync***. En la figura 1.14 se puede observar lo antes mencionado.

En la transmisión síncrona, como su nombre indica solo es posible trabajar con una señal de reloj.

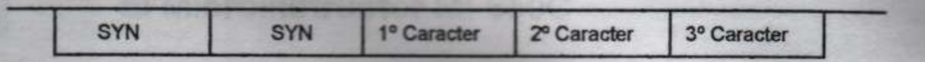


Figura 1.14. Formato de la transmisión serie síncrona

En el modo asíncrono los caracteres se transmiten de forma aislada, pudiendo variar el tiempo entre un carácter y el siguiente. El formato de transmisión se ve en la figura 1.15

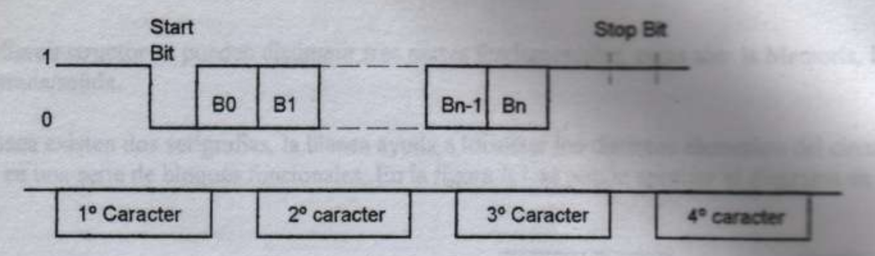


Figura 1.15. Formato de la transmisión serie asíncrona

La línea serie es inactiva a nivel alto, cuando se produce la transmisión se pone a nivel bajo, es el ***Start Bit*** o bit de arranque, su duración es fija. Al final de un carácter, la línea pasa a nivel alto durante un mínimo de 1 bit y un máximo de dos bits. Estos bits reciben el nombre de ***Stop Bit*** o bit de parada.

Los caracteres constan de un máximo de 8 bits, y se transmiten primero el bit de menos peso del carácter y luego el de más peso. De forma opcional se pueden transmitir al final del dato y antes del bit de parada, un ***bit de paridad***, para comprobar si ha habido fallos en al transmisión.

Tema 2 El Microinstructor MI-650C

2.1 Diagrama de Bloques

El Microinstructor MI - 650C es un microcomputador basado en una version CMOS del microprocesador de 8 bits 6502.

En el Microinstructor se pueden distinguir tres partes fundamentales, estas son: la Memoria, la CPU y los elementos de entrada/salida.

En la placa existen dos serigrafias, la blanca ayuda a localizar los distintos elementos del circuito, la amarilla divide el equipo en una serie de bloques funcionales. En la figura 2.1 se puede apreciar el diagrama en bloques del Microinstructor.

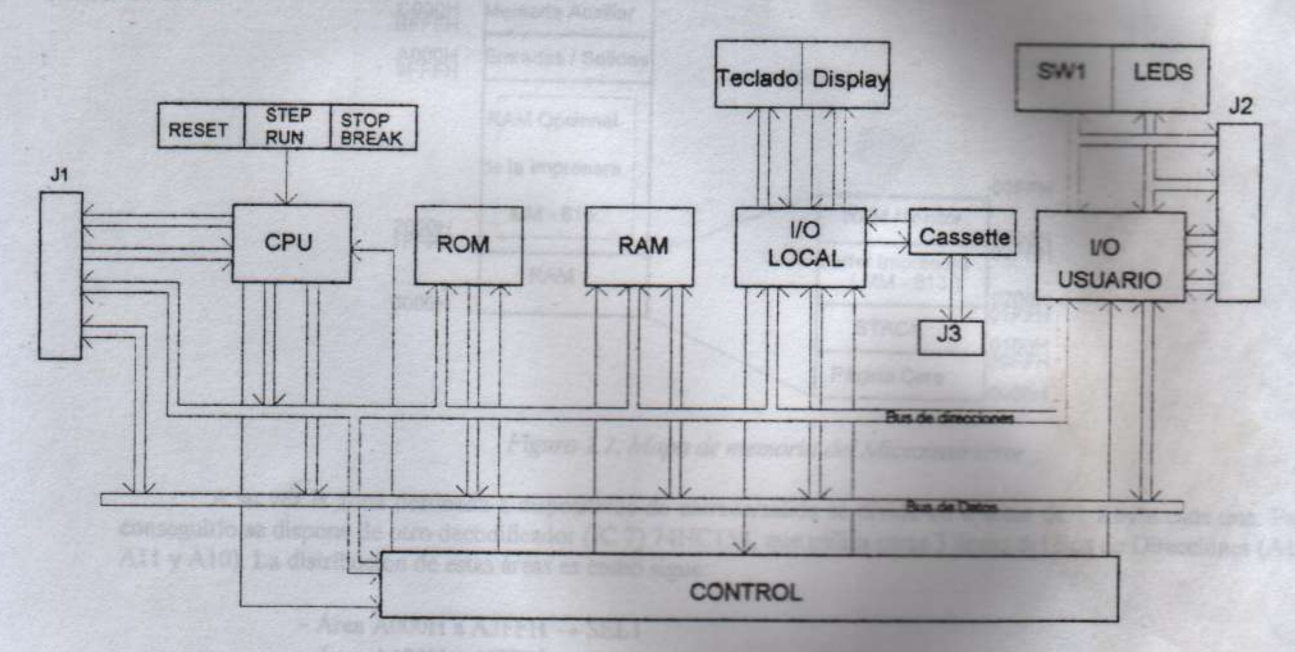


Figura 2.1 Diagrama en bloques del Microinstructor MI-650C.

2.2 Mapa de Memoria del Microinstructor

El Microinstructor se suministra con 8 Kbytes de memoria RAM y 8 Kbytes de memoria ROM.

En la memoria EPROM 2764 (IC17) suministrada con el equipo, se encuentra el programa monitor, encargado de controlar el equipo permanentemente desde que se conecta hasta que se transfiere el control a un programa de usuario. También en esta memoria se encuentran grabadas algunas prácticas y los programas que se proponen como ejemplo de los Módulos de Aplicación del Microinstructor.

La capacidad de memoria puede ser ampliada a través del zócalo de memoria Auxiliar. Puede ser RAM ó EPROM y de 2 ó 8 Kbytes. Los conectores programables J9 y J6 permiten realizar la selección anterior.

El conector (J5) permite inhibir el direccionamiento del zócalo Auxiliar, facilitando, por el contrario, la selección de un zócalo exterior de memoria en la misma posición de memoria del Microinstructor. Este selector debe de estar normalmente en la posición INT. Al conectar la Impresora MM - 613, ha de pasarse a la posición EXT. Esto es debido a que el programa de gobierno en EPROM de la Impresora, coincide con la dirección de la memoria Auxiliar del Microinstructor.

Los 64 Kbytes de memoria direccionables mediante las 16 líneas del Bus de Direcciones se dividen en 8 zonas de 8 Kbytes cada una. Las 3 líneas de mayor peso del Bus de Direcciones (A13, A14 y A15), son las encargadas de realizar la decodificación, mediante el circuito integrado 74HC138 (IC1). De las 8 salidas, son usadas 4 en el equipo. Las cuatro áreas en que queda dividida la memoria son:

- Area 0000H a 1FFFH → Asignada a RAM
- Area A000H a BFFFH → Asignada a I/O
- Area C000H a DFFFH → Asignada a Memoria Auxiliar
- Area E000H a FFFFH → Asignada a ROM

En la figura 2.2 aparece reflejada la distribución de la memoria del Microinstructor.

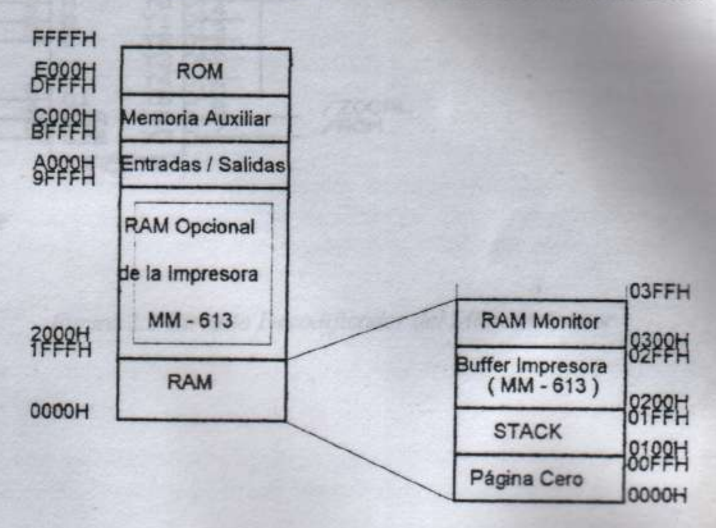


Figura 2.2. Mapa de memoria del Microinstructor

A su vez la zona destinada a dispositivos de entrada/salida se divide en 8 áreas de 1 Kbyte cada una. Para conseguirlo se dispone de otro decodificador (IC 7) 74HC158, que utiliza otras 3 líneas del Bus de Direcciones (A12, A11 y A10). La distribución de estas áreas es como sigue:

- Area A000H a A3FFH → SEL1
- Area A400H a A7FFH → SEL0
- Area A800H a ABFFH → PERI0
- Area AC00H a AFFFH → PERI1
- Area B000H a BFFFH → No se utiliza

Las dos primeras señales sirven para controlar los circuitos LSI de I/O. La señal SEL0 selecciona la VÍA 65C22 (IC22), que controla las líneas de I/O accesibles a través del conector de aplicación del Microinstructor y que actúan sobre los Leds DL4 a DL11. La señal SEL1 selecciona la VÍA 65C22 (IC19), que controla el teclado y el display del Microinstructor.

Las líneas PERI0 y PERI1, se envían a través del *Conector de Expansión* del Microinstructor, de manera que se puedan seleccionar dispositivos externos como parte de memoria del Microinstructor. En la figura 2.3 se puede ver el circuito decodificador del Microinstructor.

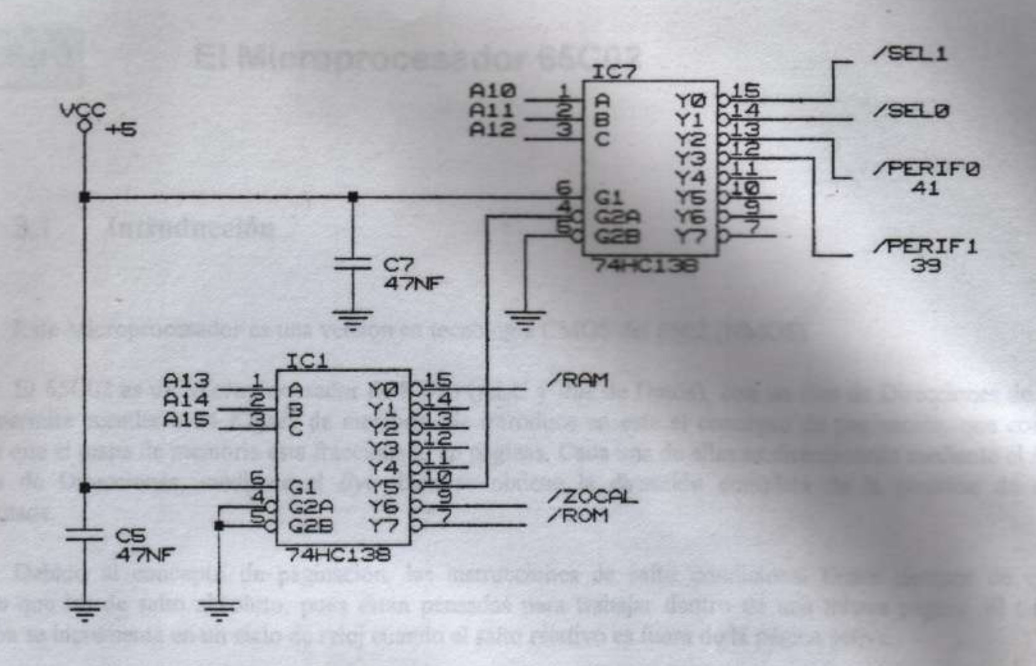


Figura 2.3 Circuito Decodificador del Microinstructor

Tema 3 El Microprocesador 65C02

3.1 Introducción

Este Microprocesador es una versión en tecnología CMOS del 6502 (NMOS).

El 65C02 es un Microprocesador de 8 *Bits* (ALU y *Bus* de Datos), con un *Bus* de Direcciones de 16 *Bits*, que le permite acceder a 64 K*bytes* de memoria. Se introduce en este el concepto de paginación, que consiste en suponer que el mapa de memoria esta fraccionado en páginas. Cada una de ellas es direccionada mediante el *Byte* alto del *Bus* de Direcciones, mediante el *Byte* bajo se obtiene la dirección completa de la posición de memoria seleccionada.

Debido al concepto de paginación, las instrucciones de salto condicional tienen tiempos de ejecución menores que las de salto absoluto, pues están pensadas para trabajar dentro de una misma página. El tiempo de ejecución se incrementa en un ciclo de reloj cuando el salto relativo es fuera de la página activa.

La "página cero" permite accesos de lectura/escritura más rápidos que en instrucciones de direccionamiento absoluto, pues corresponde a las direcciones absolutas 0000H a 00FFH. La "página 1", situada entre las posiciones 0100H y 01FFH, es la zona reservada para implementar el *Stack*, que el microprocesador trata de forma automática como memoria de acceso secuencial.

La versión CMOS de este microprocesador elimina algunas deficiencias respecto a la NMOS. Dispone de 68 mnemónicos, que junto con los distintos modos de direccionamiento da un total de 210 códigos de operación.

3.2 Estructura del hardware interno

En la figura 3.1 aparece detallado el diagrama de bloques del Microprocesador 65C02.

En este capitulo se hará una breve descripción de la CPU, ya que en el tema 1 se ha tratado.

Decodificador de instrucciones y Secuenciador

Encargado de recibir el código de la instrucción en curso desde el registro de instrucciones y datos y lo decodifica. El Secuenciador genera las señales de control pertinentes, encaminadas a la ejecución de la instrucción.

ALU y Acumulador

Realiza las operaciones lógicas y aritméticas en estrecha colaboración con el Acumulador, que proporciona uno de los operandos y almacena el resultado.

Registros índices (X e Y)

Son operativos cuando se trabaja con direccionamiento indexado, con el que la localización del operando de una instrucción se busca en la memoria añadiendo el contenido de estos registros a la dirección especificada en la instrucción.

Registro de Estado (P)

Se trata de un registro de 8 *Bits*, 7 de ellos son significativos, pues, señalan determinados sucesos al ejecutarse una instrucción. En la figura 3.2, quedan reflejados los *Bits* de este registro.

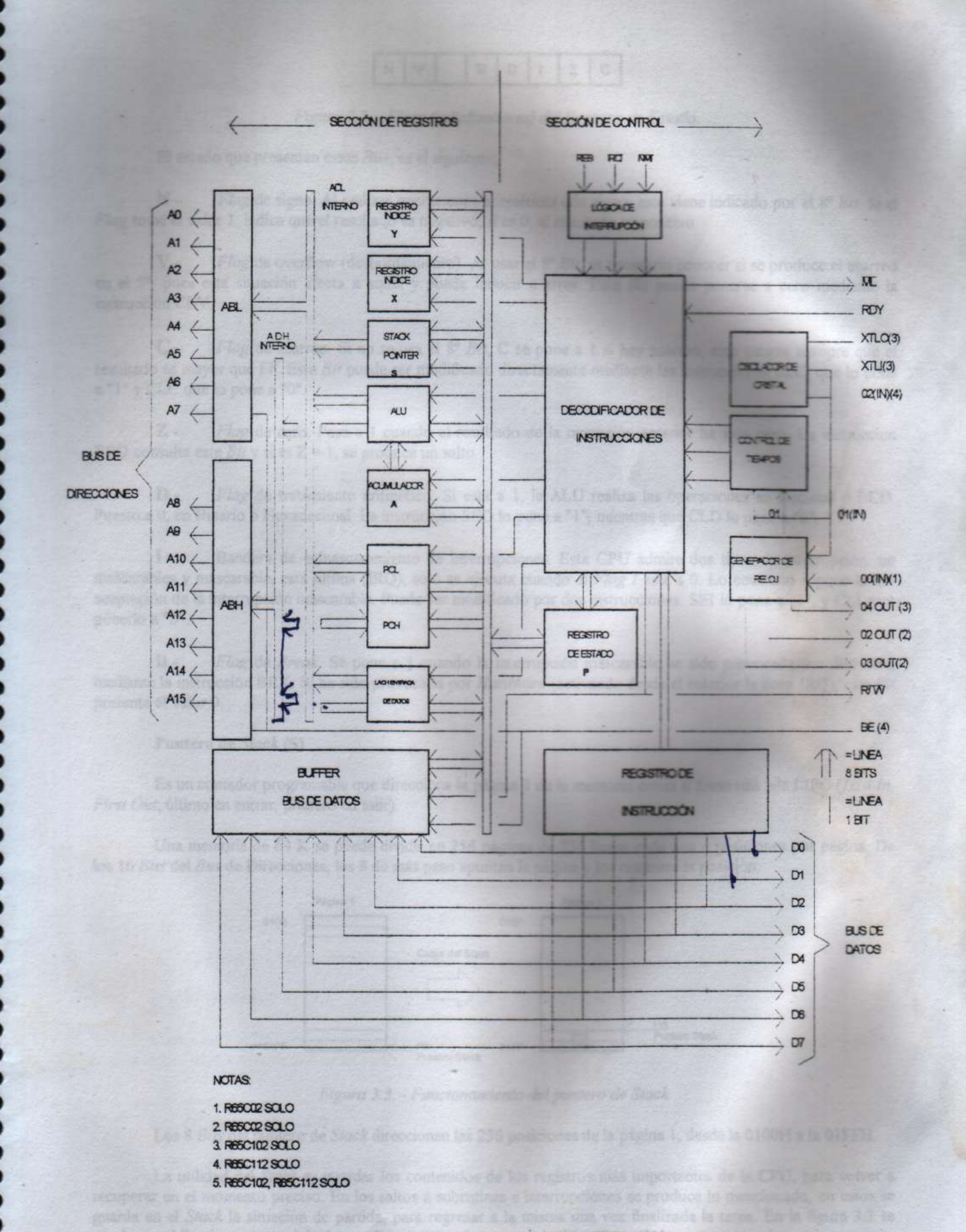


Figura 3.1. - Arquitectura general de la CPU 65C02

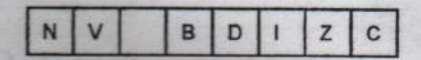


Figura 3.2. - Flags (señalizadores) del Registro de Estado.

El estado que presentan estos *Bits*, es el siguiente;

N - *Flag* de signo. Al realizar operaciones aritméticas con signo, este viene indicado por el 8° *Bit*. Si el *Flag* toma el valor 1, indica que el resultado es negativo; si es 0, el resultado es positivo.

V - *Flag* de overflow (desbordamiento). Al usar el 8° *Bit*, es necesario conocer si se produce el acarreo en el 7°, pues esta situación afecta a aquel y puede inducir a error. Este *Bit* puede ponerse a cero mediante la instrucción CLV.

C - *Flag* de acarreo. Si no se usa el 8° *Bit*, C se pone a 1 si hay acarreo, esto ocurre siempre que el resultado es mayor que FF. Este *Bit* puede ser modificado directamente mediante las instrucciones SEC, que lo pone a "1" y CLC que lo pone a "0".

Z - *Flag* de cero. Pasa a 1 cuando el resultado de la operación anterior ha sido cero. La instrucción BEQ consulta este *Bit* y si es Z = 1, se produce un salto.

D - *Flag* de tratamiento aritmético. Si esta a 1, la ALU realiza las operaciones en Decimal o BCD. Puesto a 0, en Binario o Hexadecimal. La instrucción SED lo pone a "1", mientras que CLD lo pone a "0".

I - Bandera de enmascaramiento de interrupciones. Esta CPU admite dos tipos de interrupción: no mascarables y mascarable, esta última (IRQ), sólo se ejecuta cuando el *Flag I* esta a 0. Lo contrario supone la no-aceptación de la interrupción mascarable. Puede ser modificado por dos instrucciones: SEI lo pone a "1", y CLI para ponerlo a "0".

B - *Flag* de *Break*. Se pone a 1 cuando la interrupción mascarable ha sido provocada por *Software*, mediante la instrucción BRK. Si ha sido provocada por *Hardware* (activando desde el exterior la linea IRQ), este *Bit* presenta el valor 0.

Puntero de *Stack* (S)

Es un contador programable que direcciona la página 1 de la memoria como si fuese una pila LIFO (*Last In, First Out*; último en entrar, primero en salir).

Una memoria de 64 K se puede dividir en 256 páginas de 256 lineas cada una o posiciones por página. De los 16 *Bits* del *Bus* de Direcciones, los 8 de más peso apuntan la página y los restantes la posición.

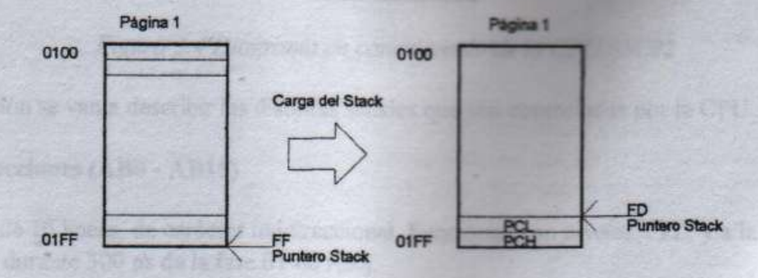


Figura 3.3. - Funcionamiento del puntero de Stack

Los 8 *Bits* del puntero de *Stack* direccionan las 256 posiciones de la página 1, desde la 0100H a la 01FFH.

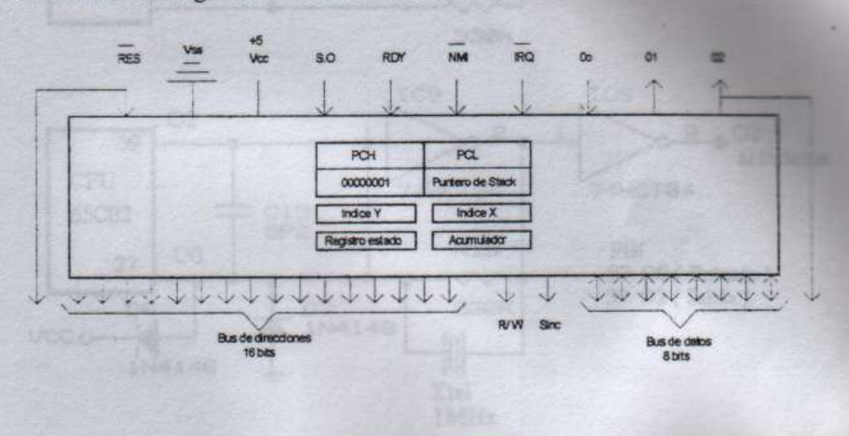
La utilidad del *Stack* es guardar los contenidos de los registros más importantes de la CPU, para volver a recuperar en el momento preciso. En los saltos a subrutinas e interrupciones se produce lo mencionado, en estos se guarda en el *Stack* la situación de partida, para regresar a la misma una vez finalizada la tarea. En la figura 3.3 se puede ver el funcionamiento del mismo, el puntero se decrementa cada vez que algún registro se salva en la memoria. La devolución de los registros se efectúa en forma inversa a su almacenamiento (LIFO).

Contador de Programa (PC)

Es un contador de 16 *Bits*, los 8 *Bits* de más peso (PCH) seleccionan la página y los 8 de menos peso (PCL) seleccionan la posición. La misión del PC es de enviar por el *Bus* de direcciones, la dirección efectiva de memoria donde se encuentran los datos relativos al código de la instrucción en curso.

3.3 Diagrama de conexionado del μP65C02

La figura 3.4 muestra el diagrama de conexionado de esta CPU.



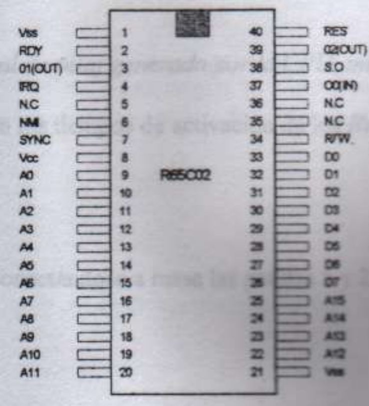


Figura 3.4 Diagrama de conexionado de la CPU 65C02

A continuación se van a describir las distintas señales que son controladas por la CPU.

Bus de Direcciones (AB0 - AB15)

Compuesto de 16 líneas, de carácter unidireccional. Funcionan con niveles TTL, y a la frecuencia de 1 MHz, la dirección es válida durante 300 ns de la fase θ1 de reloj.

Bus de Datos (DB0 - DB7)

Consta de 8 líneas bidireccionales, por las que se transfieren los datos y las instrucciones. En su salida existen *Buffers* amplificadores triestado. Estos permanecen en estado flotante, a excepción del tiempo en que se transmiten datos. A la frecuencia de 1 MHz los datos permanecen estables en el *Bus*, 100 ns antes de finalizar la fase θ2.

Señales de reloj (θ0, θ1 y θ2)

La CPU dispone de un generador interno de señales de reloj y sincronismo, saliendo O1 por la patilla 3 y O2 por la 39. Para estabilizar la frecuencia requiere de una señal O0 desde el exterior, que se introduce por la patilla 37. La señal estabilizada se puede generar mediante un cristal de cuarzo o mediante una red RC. O2 requiere una doble inversión para conformar la onda de forma adecuada, además de introducir el necesario retardo, ver figura 3.5.

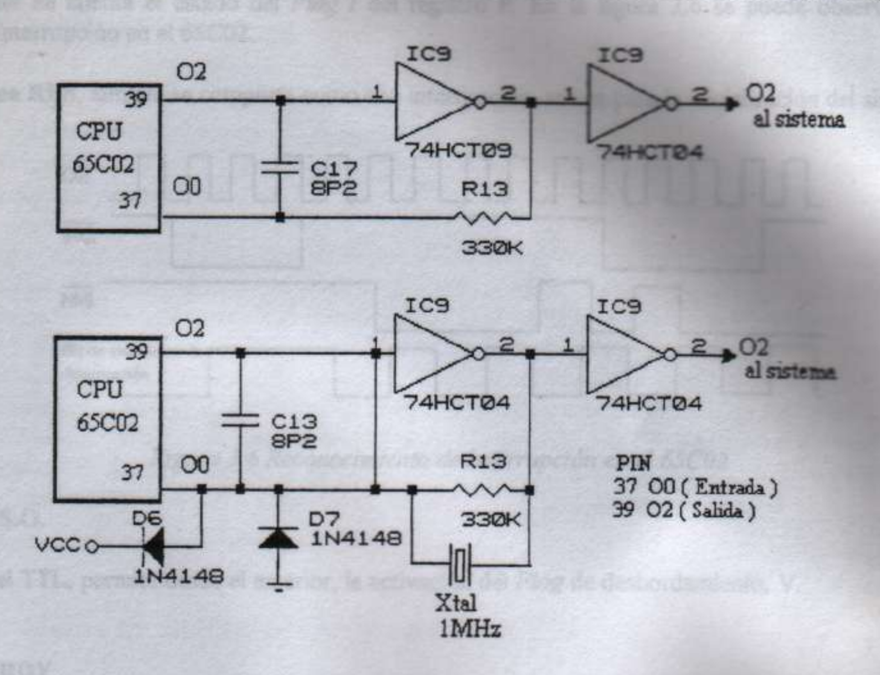


Figura 3.5. - Estabilización de la señal de reloj generada por la CPU, mediante red RC o por cristal de cuarzo.

Las señales O1 y O2 determinan los tiempos de activación de los *Buses* en los ciclos de lectura y escritura.

Línea de alimentación

Se aplican +5 V a la patilla 8, conectándose a masa las patillas 5 y 21.

Líneas de interrupción

Esta CPU dispone de 3 líneas, por las que desde el exterior es posible provocar diferentes tipos de interrupciones, siendo activas a nivel bajo (IRQ, NMI Y RES).

Las interrupciones que se pueden provocar tienen carácter vectorizado, cada una de ellas tiene un "vector de interrupción", compuesto por dos posiciones específicas de memoria en las que se introduce la dirección de inicio del programa que atiende a la interrupción. La interrupción NMI esta posicionada en FFFAH y FFFBH, RES está entre FFFCH y FFFDH, y la IRQ entre FFFEH y FFFFH.

Las interrupciones son tratadas del siguiente modo:

1°) Activada la línea de interrupción, y caso de ser mascarable (IRQ), se consulta el *Flags I* del registro P, para ver si es admitida.

2°) Una vez admitida la interrupción, la instrucción en curso finaliza, salvándose los contenidos del PC y del registro de estado en la zona de memoria controlada por el puntero de *Stack*.

3°) El contenido del vector de interrupción correspondiente a la interrupción activada, compuesto por 16 *Bits*, forma una dirección de memoria que se carga en el PC, trasladándose por el *Bus* de Direcciones al programa que se ha establecido para la atención de dicha interrupción.

4º) Finalizado el programa de atención a la interrupción, la última instrucción RTI (Retorno de subrutina), recupera desde la memoria del *Stack* la posición de salida del programa.

La línea /NMI no es interrupción mascarable, por lo que siempre se efectúa. Obliga a alterar el contenido del PC, sin tener en cuenta el estado del *Flag I* del registro P. En la figura 3.6 se puede observar la forma de reconocer una Interrupción en el 65C02.

La línea RES, aunque se comporta como una interrupción, se usa para la inicialización del sistema.

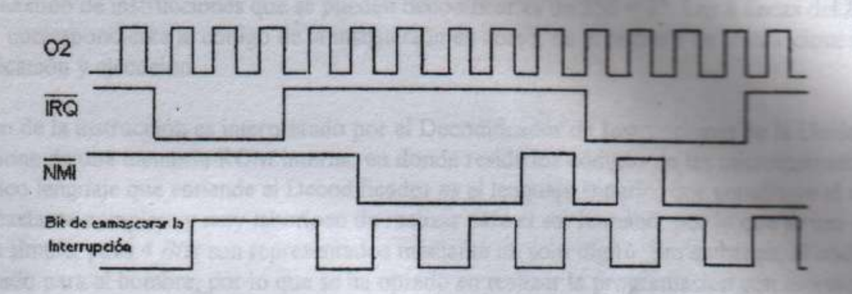


Figura 3.6 Reconocimiento de Interrupción en el 65C02

Línea S.O.

De nivel TTL, permite desde el exterior, la activación del *Flag* de desbordamiento, V.

Línea RDY

Adapta la CPU a la menor velocidad de las memorias y al "Acceso directo a memoria" (DMA). La señal de RDY detiene a la CPU en todos los ciclos, excepto en los de escritura, y deja en estado flotante a los *Buses* de direcciones y datos.

Línea de R/$\overline{W}$

Sólo pasa a nivel bajo cuando se realiza una escritura de datos, ya sea en la memoria o en los periféricos exteriores.

Línea de SYNC

Es una línea de salida de la CPU, que identifica en el exterior, los ciclos en los que se realiza la búsqueda (*Fetch*), de un código de operación (OP), Pasa a nivel alto durante la fase θ1 de la búsqueda del código de operación de una instrucción. En la figura 3.7 queda detallado el proceso de activación de esta línea.

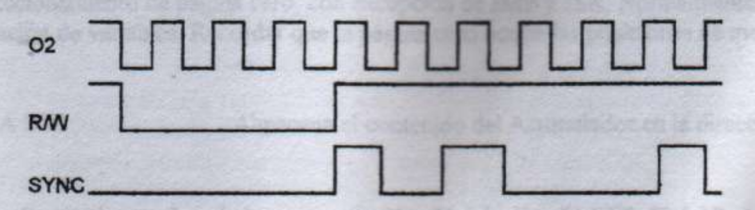


Figura 3.7 Señal SYNC del 65C02

Tema 4 Programación del µP 65C02

4.1. El lenguaje maquina

Las instrucciones van desde la memoria de programa hasta la CPU, por el *Bus* bidireccional de datos. Al ser este de 8 *Bits* el máximo de instrucciones que se pueden decodificar es de $256 = 2^8$. Las 8 líneas del *Bus* de datos depositan 1 *Byte*, correspondiente al código de la instrucción en curso, en el registro de Instrucciones y datos, para su posterior decodificación y ejecución.

El código de la instrucción es interpretado por el Decodificador de Instrucciones de la Unidad de Control de la CPU. Este dispone de una memoria ROM interna, en donde reside los códigos de las microinstrucciones asociadas a la misma. El único lenguaje que entiende el Decodificador es el lenguaje Binario, que constituye el *código maquina*. Este lenguaje es bastante complejo y muy laborioso de realizar para el ser humano, por lo que se usa el código hexadecimal, más simple, pues 4 *Bits* son representados mediante un solo dígito. Sin embargo, el código hexadecimal carece de significado para el hombre, por lo que se ha optado en realizar la programación con *mnemónicos*. La programación en mnemónico esta formada por tres letras que se corresponde con las iniciales del nombre de la instrucción en Ingles. Un ejemplo típico es la instrucción " Load Acumulator", que equivale en español a la "Carga del Acumulador", las iniciales son: LDA y equivale a A9 en Hexadecimal, siendo: 1010 1001.

4.2. Modos de direccionamiento en el µP65C02

Según el modo que siga una instrucción para definir o situar al operando que le afecta, su código máquina varía, lo que supone que una misma instrucción pueda adoptar diversos códigos, de acuerdo con el direccionamiento del operando.

El 65C02 dispone de un total de 68 mnemónicos, que expandidos a los 15 modos de direccionamiento da un total de 210 códigos de operación. El mismo microprocesador 6502 en tecnología NMOS sólo admite 56 mnemónicos y 151 códigos de operación con 9 modos de direccionamiento.

Los modos de direccionamiento admitidos por el 65C02 son:

1- Direccionamiento Inmediato (#nn): El valor del operando manejado por la instrucción, se coloca detrás del *Byte* que expresa el código de operación de la misma. Usa 2 *Bytes*, una relativa al código de operación y otra para el operando.

Ejemplo: LDA #2FH — Carga el Acumulador con el dato (0010 1111B).

2- Direccionamiento de página cero (ZP): El operando de la instrucción está contenido en la posición que se especifica en la página cero. Precisa de 2 *Bytes*. Todas las instrucciones válidas en direccionamiento absoluto lo son también en direccionamiento de página cero, con excepción de JMP y JSR. Normalmente la página cero se reserva para la declaración de variables. Recordar que la página cero ocupa las posiciones de memoria 00H hasta FFH.

Ejemplo: STA 20H — Almacena el contenido del Acumulador en la dirección 20H de página cero.

3 y 4- Direccionamiento de página cero, índice X ó índice Y (ZP, X ó Y): Las instrucciones de este tipo incluyen un sólo operando que constituye la dirección base de página cero a la que se sumará el registro de indice. Según el valor de este índice se puede acceder a cualquier dirección de página cero.

Ejemplo: LDA 10H,X — Cuando X contiene 25H, transfiere al Acumulador el dato contenido en la dirección de página cero 35H.

5- Direccionamiento Absoluto (ABS): Compuesta por 3 *Bytes*, uno de código de operación y dos de operando. Los dos *Bytes* que siguen al código de operación especifican la dirección absoluta de memoria en la que se encuentra el dato a procesar. Esta dirección esta almacenada en memoria en orden inverso, primero el *Byte* bajo y segundo el *Byte* alto.

Ejemplo: LDX 1234H Carga el registro X con el dato que esta en la dirección 1234H.

6 y 7- Direccionamiento absoluto, índice X ó índice Y (ABS, X ó Y): Es una variante del direccionamiento absoluto. Se puede efectuar bien con el registro X ó con el Y. El valor efectivo de la dirección de memoria se obtiene sumando la dirección absoluta, indicada en los *Bytes* siguientes al código de operación, con el valor que en ese momento tenga el registro índice usado. Como el registro X e Y son de 1 *Byte*, con este direccionamiento se puede acceder a cualquier dirección de un bloque de memoria de 256 *Bytes*.

Ejemplo: LDA 1000H,X Cuando X contiene 25H, transfiere al Acumulador el dato contenido en la dirección 1025H.

8- Direccionamiento indexado, indirecto, X ((IND, X)): Las instrucciones que lo usan constan de dos *Bytes*. El primero indica el código de operación, el segundo representa una dirección de página cero, que se suma al contenido del registro índice X. El resultado de esta suma apunta a una posición de página cero, que contiene los 8 *Bits* de menos peso de la dirección efectiva donde se encuentra el dato. La siguiente posición de página cero contiene los 8 *Bits* de más peso de la dirección efectiva.

Ejemplo: LDA (20H,X) Si el registro X contiene 08H y en las posiciones 0028H y 0029H se encuentran respectivamente 34H y 12H, en la dirección 1234H se encuentra el dato 55H; al ejecutar esta instrucción el Acumulador se cargará con el dato 55H.

9-.Direccionamiento indirecto, indexado, Y ((IND, Y)): El segundo *Byte* de esta instrucción apunta a una posición de página cero, a cuyo contenido se suma el del registro índice Y, siendo el resultado de la suma los 8 *Bits* de menos peso de la dirección efectiva de la memoria, donde se encuentra el operando. El *Carry* de esta suma, si lo hay, se suma al contenido de la siguiente posición de la página cero y así se consiguen los 8 bits de más peso de la dirección efectiva.

Ejemplo: LDA (50H,Y) Si la posición de memoria 50H, de página cero, contiene el dato 24H, y en la posición siguiente, 51H, hay 12H; el registro Y contiene 10H. El resultado de sumar el contenido de la posición de memoria indicada en el operando con el registro X, seria 34H. En la dirección 1234H se encuentra el dato 55H; al ejecutar esta instrucción el Acumulador se cargará con el dato 55H.

10- Direccionamiento de Acumulador (AC): La instrucción sólo implica la participación del Acumulador. Tiene 1 sólo *Byte* de longitud.

Ejemplo: DEC A Decrementa el Acumulador y deja el resultado en el mismo.

11- Direccionamiento relativo (REL): Utilizado exclusivamente por instrucciones de salto condicional (*Branch*). La dirección de memoria a la que se transfiere el control del programa se obtiene sumando el valor del contador de programa al segundo *Byte* de la instrucción, denominado "*Offset*". El *Offset* es un número en complemento a 2, con lo que se pueden efectuar saltos de hasta 127 posiciones hacia adelante o 128 hacia atrás.

Ejemplo: BNE 20H Si la instrucción esta contenida en las posiciones 1000H y 1001H del programa, el valor del PC actualizado será 1002H. Si el Flag Z del registro de estado es distinto de 0 el valor del "*Offset*" se sumará al del PC y la ejecución del programa continuará a partir de la dirección 1022H de memoria. En caso contrario la siguiente instrucción a ejecutar será la que ocupe la dirección 1002H.

12- Direccionamiento absoluto indirecto (ABS): Sólo afecta a la instrucción de salto incondicional JMP. En la que el segundo y tercer *Byte* de la instrucción indican los 2 *Bytes* que forman la nueva dirección que se carga en el contador de programa.

Ejemplo: JMP 1234H La dirección indicada, 1234H, se carga en el PC, y, por tanto, se cede el control del programa a la instrucción que se encuentre en dicha dirección.

13- Direccionamiento implícito: Es propio de instrucciones que afectan a un solo registro. El operando esta implicado en la propia instrucción. 29 instrucciones usan este tipo de direccionamiento. Tiene 1 *Byte* de longitud.

Ejemplo: TAX Transfiere el Acumulador al Registro X.

4.3. Juego de instrucciones

En este apartado se mostraran agrupadas todas las instrucciones del μP65C02, según el modo de operación.

Instrucciones de Movimiento de Datos					
Transferencia entre Registros		Transferencia entre Registros y Stack		Transferencia entre Registros y Memoria	
TAX	$A \rightarrow X$	PHA	$A \rightarrow SP$	LDA	$M \rightarrow A$
TXA	$X \rightarrow A$	PLA	$SP \rightarrow A$	STA	$A \rightarrow M$
TAY	$A \rightarrow Y$	PHP	$P \rightarrow SP$	LDX	$M \rightarrow X$
TYA	$Y \rightarrow A$	PLP	$SP \rightarrow P$	STX	$X \rightarrow M$
TSX	$SP \rightarrow X$	PHX	$X \rightarrow SP$	LDY	$M \rightarrow Y$
		PLX	$SP \rightarrow X$	STY	$Y \rightarrow M$
		PHY	$Y \rightarrow SP$	STZ	$0 \rightarrow M$
		PLY	$SP \rightarrow Y$		

Instrucciones Aritmético – Lógicas				**Instrucciones de Comparación**	
Operaciones Aritméticas		Operaciones Lógicas			
ADC	$A + M + C \rightarrow A$	AND	$A \bullet M \rightarrow A$	CMP	¿ $A = M$?
INC	$M + 1 \rightarrow A$	ORA	$A \vee M \rightarrow A$	CPX	¿ $X = M$?
INX	$X + 1 \rightarrow A$	EOR	$A \oplus M \rightarrow A$	CPY	¿ $Y = M$?
INY	$Y + 1 \rightarrow A$			$C = 0 \rightarrow R < M$	
SBC	$A - M - \overline{C} \rightarrow A$			$C = 1 \rightarrow R \geq M$	
DEC	$M - 1 \rightarrow A$			$Z = 0 \rightarrow R \neq M$	
DEX	$X - 1 \rightarrow A$			$Z = 1 \rightarrow R = M$	
DEY	$Y - 1 \rightarrow A$				

Instrucciones de Desplazamiento y Rotación		**Instrucciones de Salto**			
		Salto Incondicional		Salto condicional	
ASL	Despl. $M \rightarrow$ Izquierda	BRA	Relativo	BCC	Sí $C = 0$
LSR	Despl. $M \rightarrow$ Derecha	JMP	Absoluto	BCS	Sí $C = 1$
ROL	Rota $M \rightarrow$ Izquierda	JSR	Subrutina	BEQ	Sí $Z = 1$
ROR	Rota $M \rightarrow$ Derecha	BRK	Subrutina de Break	BNE	Sí $Z = 0$
ASL	Despl. $A \rightarrow$ Izquierda	RTS	Volver de Subrutina	BPL	Sí $N = 0$
LSR	Despl. $A \rightarrow$ Derecha	RTI	Volver de Subr. Inter	BMI	Sí $N = 1$
ROL	Rota $A \rightarrow$ Izquierda			BVC	Sí $V = 0$
ROR	Rota $A \rightarrow$ Derecha			BVS	Sí $V = 1$
				BBRn	Sí *Bit* n de $M = 0$
				BBSn	Sí *Bit* n de $M = 1$

Instrucciones de modificación de flags		**Instrucciones de Test de *Bits* y Manipulación**		**Instrucción Nula**	
CLC	$C = 0$	BIT	Test *Bits* M y A	NOP	No operación
SEC	$C = 1$	RMBn	Pone "0" *Byte* "n" de M		
CLD	$D = 0$ Modo Binario	SMBn	Pone "1" *Byte* "n" de M		
SED	$D = 1$ Modo Decimal	TRB	Test de M con A y pone "0"		
CLI	$I = 0$ Habilita /IRQ	TSB	Test de M con A y pone "1"		
SEI	$I = 1$ Inhibe /IRQ				
CLV	$V = 0$				

4.4. Juego de Instrucciones del μP65C02

Mnemónico y función	Direccionamiento	Código OP	Número bytes	Número ciclos
1º) ADC	Inmediato	69	2	2
	Página cero	65	2	3
	Página cero, X	75	2	4
	Absoluto	6D	3	4
	Absoluto, X	7D	3	4
A+(M)+C→A,C	Absoluto, Y	79	3	4
	Indirecto, X	61	2	6
	Indirecto, Y	71	2	5
	Indirecto mejorado	72		
2º) AND	Inmediato	29	2	2
A • (M)→A	Página cero	25	2	3
	Página cero, X	35	2	4
	Absoluto	2D	3	4
	Absoluto, X	3D	3	4
	Absoluto, Y	39	3	4
	Indirecto, X	21	2	6
	Indirecto, Y	31	2	5
	Indirecto mejorado	32		
3º) ASL	Acumulador	0A	1	2
	Página cero	06	2	5
C ← [7 0] ← 0	Página cero, X	16	2	6
	Absoluto	0E	3	6
	Absoluto, X	1E	3	7
4º) BBR. Branch si n = 0	Página cero	nF, n entre 0 y 7		5+
5º) BBS Branch si bit N = 1 → (M)	Página cero	mF, m = Nº H entre 8 y F. m = N + 8		5+
6º) BCC. Branch si C=0. Offset.	Relativo	90	2	2
7º) BCS. Branch si C=1. Offset.	Relativo	B0	2	2
8º) BEQ. Branch si Z=1. Offset.	Relativo	F0	2	2
9º) BIT	Página cero	24	2	3
A ∧ (M)	Absoluto	2C	3	4
	Inmediato	89		2
	Absoluto, X	3C		4
	Página cero	34		4
10º) BMI. Branch si N=1. Offset.	Relativo	30	2	2
11º) BNE. Branch si Z=0. Offset	Relativo	D0	2	2
12º) BPL. Branch si N=0. Offset.	Relativo	10	2	2
13º) BRA. Salto incondicional	Relativo	80		3
14º) BRK. Salto a subrutina de interrupción	Implicado	00	1	7
15º) BVC. Branch si V=0. Offset.	Relativo	50	2	2
16º) BVS. Branch si V=1. Offset.	Relativo	70	2	2
17º) CLC. C ← 0	Implicado	18	1	2
18º) CLD. D ← 0	Implicado	D8	1	2
19º) CLI. I ← 0	Implicado	58	1	2
20º) CLV. V ← 0	Implicado	B8	1	2
21º) CMP	Inmediato	C9	2	2
A - (M)	Página cero	C5	2	3
	Página cero, X	D5	2	4
	Absoluto	CD	3	4
	Absoluto, X	DD	3	4
	Absoluto, Y	D9	3	4
	Indirecto, X	C1	2	6
	Indirecto, Y	D1	2	5
	Indirecto mejorado	D2		5
22º) CPX	Inmediato	E0	2	2
X - (M)	Página cero	E4	2	3
	Absoluto	EC	3	4
23º) CPY	Inmediato	C0	2	2
Y - (M)	Página cero	C4	2	3
	Absoluto	CC	3	4
24º) DEC (M) - 1 → M	D Flag modo decimal	(M) Contenido de memoria	S Stack	
25º) DEX. (x) - 1 → x	Implicado	CA	1	2
26º) DEY. (Y) - 1 → Y	Implicado	88	1	2
27º) EOR	Inmediato	49	2	2
A ⊕ (M) → A	Página cero	45	2	3
	Página cero, X	55	2	4
	Absoluto	4D	3	4
	Absoluto, X	5D	3	4
	Absoluto, Y	59	3	4
	Indirecto, X	41	2	6
	Indirecto, Y	51	2	5
	Indirecto mejorado	52	2	5
28º) INC	Acumulador	1A	2	5
(M) + 1 → A	Página cero	E6	2	6
	Página cero, X	F6	3	6
	Absoluto	EE	3	7
	Absoluto, X	FE		7
29º) INX. (X) + 1 → X	Implicado	E8	1	2
30º) INY. (Y) + 1 → Y	Implicado	C8	1	2
31º) JMP	Absoluto	4C	3	3
Salto a nueva dirección	Indirecto	6C	3	5
	Indirecto, X	7C		6
32º) JSR. Salto a subrutina	Absoluto	20	3	6
33º) LDA	Inmediato	A9	2	2
(M) → A	Página cero	A5	2	3
	Página cero, X	B5	2	4
	Absoluto	AD	3	4
	Absoluto, X	BD	3	4*
	Absoluto, Y	B9	3	4*
	Indirecto, X	A1	2	6
	Indirecto, Y	B1	2	5*
	Indirecto mejorado	B2		5
34º) LDX	Inmediato	A2	2	2
(M) → X	Página cero	A6	2	3
	Página cero, Y	B6	2	4
	Absoluto	AE	3	4
	Absoluto, Y	BE	3	4
35º) LDY	Inmediato	A0	2	2
Y ← (M)	Página cero	A4	2	3
	Página cero, X	B4	2	4
	Absoluto	AC	3	4
	Absoluto, X	BC	3	4
36º) LSR	Página cero	46	2	5
	Acumulador	4A	1	2
0 → [7 0] → C	Página cero, X	56	2	6
	Absoluto	4E	3	6
	Absoluto, X	5E	3	7
37º) NOP. No hace nada	Implicado	EA	1	2
38º) ORA	Inmediato	09	2	2
A ∨ (M) → A	Página cero	05	2	3
	Página cero, X	15	2	4
	Absoluto	0D	3	4
	Absoluto, X	1D	3	4
	Absoluto, Y	19	3	4
	Indirecto, X	01	2	6
	Indirecto, Y	11	2	5
	Indirecto mejorado	12		5
39º) PHA. A → S; SP - 1	Implicado	48	1	3
40º) PHP. P → S; SP – 1	Implicado	08	1	3
41º) PHX. X → S; SP - 1	Implicado	DA		3
42º) PHY. Y → S; SP – 1	Implicado	5A		3
43º) PLA. S → A; SP + 1	Implicado	68	1	4
44º) PLP. P → S; SP + 1	Implicado	28	1	4
45º) PLX. S → X; SP + 1	Implicado	FA		4
46º) PLY. S → Y; SP + 1	Implicado	7A		4
47º) RMB Pone a 0 bit de (M)	Implicado	n7, n = Nº H entre 0 y 7		5
48º) ROL	Acumulador	2A	1	2
	Página cero	26	2	5
← [7 0] ← [C] ←	Página cero, X	36	2	6
	Absoluto	2E	3	6
	Absoluto, X	3E	3	7
49º) ROR	Acumulador	6A	1	2
	Página cero	66	2	5
→ [7 0] → [C] →	Página cero, X	76	2	6
	Absoluto	6E	3	6
	Absoluto, X	7E	3	7
50º) RTI. Retorno Interrupción	Implicado	40	1	6
51º) RTS. Retorno subrutina	Implicado	60	1	6
52º) SBC	Inmediato	E9	2	2
	Página cero	E5	2	3
	Página cero, X	F5	2	4
$A - (M) - \overline{C} \rightarrow A$	Absoluto	ED	3	4
	Absoluto, X	FD	3	4
	Absoluto, Y	F9	3	4
	Indirecto, X	E1	2	6
	Indirecto, Y	F1	2	5
	Indirecto mejorado	F2	2	5
53º) SEC. 1 → C	Implicado	38	1	2
54º) SED. D → 1	Implicado	F8	1	2
55º) SEI. I → 1	Implicado	78	1	2
56º) SMB Bit N → (M) = 1	Implicado	m7, m = Nº H entre 8 y F. m = N - 8		5+
57º) STA	Página cero	85	2	3
A → (M)	Página cero, X	95	2	4
	Absoluto	8D	3	4
	Absoluto, X	9D	3	5
	Absoluto, Y	99	3	5
	Indirecto, X	81	2	6
	Indirecto, Y	91	2	6
	Indirecto mejorado	92		5
58º) STX	Página cero	86	2	3
X → (M)	Página cero, Y	96	2	4
	Absoluto	8E	3	4
59º) STY	Página cero	84	2	3
Y → (M)	Página cero, X	94	2	4
	Absoluto	8C	3	4
60º) STZ	Página cero	64		3
Set 0 → (M)	Página cero, X	74		4
	Absoluto	9C		4
	Absoluto, X	9E		5
61º) TAX. A → X	Implicado	AA	1	2
62º) TAY. A → Y	Implicado	A8	1	2
63º) TRB	Página cero	14		5
Set 0 → (M – 1)	Absoluto	1C		6
64º) TSB	Página cero	04		5
Set 1 → (M – 1)	Absoluto	0C		6
65º) TSX. S → X	Implicado	BA	1	2
66º) TXA. X → A	Implicado	8A	1	2
67º) TXS. X → S	Implicado	9A	1	2
68º) TYA. Y → A	Implicado	98		2

Notas:

\+ Instrucción que no se puede usar en la versión NMOS

A Acumulador
N Flag de signo
V Flag de overflow
B Flag de Break
I Flag de Interrupción
Z Flag de cero
C Flag de acarreo
M Posición de memoria
• Operación AND
⊕ Operación EXOR
∨ Operación OR
SP Puntero de Stack
P Registro de estado
X Registro x
Y Registro y

Tema 5 Organigramas

5.1. Metodología de la programación

Debido a la naturaleza del ordenador, la preparación del trabajo que debe ser ejecutado por el mismo, requiere una perfecta ordenación y cuidado hasta en los últimos detalles: se necesita un MÉTODO.

Las etapas que deben seguirse en cualquier trabajo de programación, son:

a) Representación gráfica de las operaciones a realizar por el programa.
b) Codificación en lenguaje simbólico y obtención del programa fuente.
c) Preparación de un juego de datos para poder probar el programa.
d) Traducción del programa a lenguaje máquina con la obtención del programa objeto.
e) Análisis de la traducción para detectar los posibles errores de gramática que se pueden haber cometido en el programa fuente.
f) Pruebas del programa, ensayando su explotación con el juego de datos preparado anteriormente.
g) Análisis de las pruebas para detectar posibles errores en la programación.
h) Documentación del programa para su mantenimiento incluyendo instrucciones para que pueda ser ejecutado correctamente.

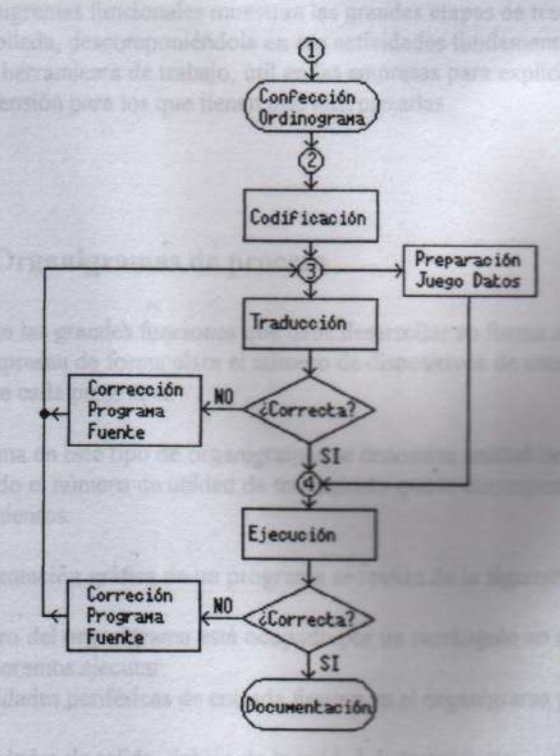


Figura 5.1 Método para la confección de un programa.

5.2. Necesidad de la representación gráfica

Se necesita una representación clara y detallada que refleje la secuencia en que deben ser ejecutadas las operaciones. El mejor ejemplo de esta necesidad lo encontramos en el punto anterior, en el que se describe un método para la confección de los programas, que, si estuviese acompañado de una representación gráfica, la comprensión sería mayor y más fácil de adquirir. La figura anterior, Figura 5.1, completa, aclara y facilita la comprensión de las distintas etapas.

La primera etapa que debemos realizar para la confección de un programa, debe ser la representación gráfica de la solución del problema que queremos mecanizar, dando lugar a los organigramas: "Representación gráfica de la estructura lógica y operacional de los problemas de ordenador".

Existen tres tipos diferentes:

ORGANIGRAMA FUNCIONAL: Muestra las grandes etapas de transformación que sufre la información sin referirse a ningún elemento del ordenador.

ORGANIGRAMA DE PROCESO: Tiene las mismas características que el anterior, pero además tiene en cuenta los elementos que constituyen el ordenador.

ORGANIGRAMA DE DETALLE U ORDINOGRAMA: Recoge gráficamente todas las órdenes que se deben seguir en secuencia para la solución del problema.

5.3. Organigramas funcionales

Los organigramas funcionales muestran las grandes etapas de transformación de cualquier acción, tal como debe de ser desarrollada, descomponiéndola en sus actividades fundamentales, en secuencia de ejecución, constituyendo una herramienta de trabajo, útil en las empresas para explicar la ejecución de las tareas, por su gran facilidad de comprensión para los que tienen que interpretarlas.

5.4. Organigramas de proceso

No reflejan las grandes funciones que debe desarrollar en forma automática el sistema de proceso de datos, pero en cambio, expresan de forma clara el número de dispositivos de entrada y de salida que deben estar disponibles para la ejecución de cada programa.

El programa en este tipo de organigramas se denomina unidad de tratamiento, y se representa mediante un rectángulo indicando el número de unidad de tratamiento que le corresponde, es decir, el número del programa que realiza estos tratamientos.

La representación gráfica de un programa se realiza de la siguiente forma:

a) El centro del organigrama está ocupado por un rectángulo en el que se indica el número de la unidad de tratamiento que queremos ejecutar.
b) Las unidades periféricas de entrada figuran en el organigrama por encima del rectángulo de la unidad de tratamiento.
c) Las unidades de salida, debajo de la unidad de tratamiento.
d) Las unidades de entrada/salida a los lados. Estas unidades periféricas tienen su representación gráfica, de tal forma que mediante sus símbolos y flechas, podemos ver con toda claridad las necesidades del ordenador para poder ejecutar cualquier programa.

5.5. Organigramas de detalle u ordinogramas

Los ordinogramas u organigramas de órdenes o de detalles, dan una representación gráfica de la estructura lógica y operacional de los problemas con el grado de detalle necesario para que, una vez confeccionados, se pueda realizar la etapa siguiente de la programación, la codificación.

Dependiendo del nivel del lenguaje simbólico utilizado, orientado a la máquina u orientado al problema, varía el grado de detalle con el que es necesario representar gráficamente las distintas órdenes.

La simbologia empleada es la siguiente:

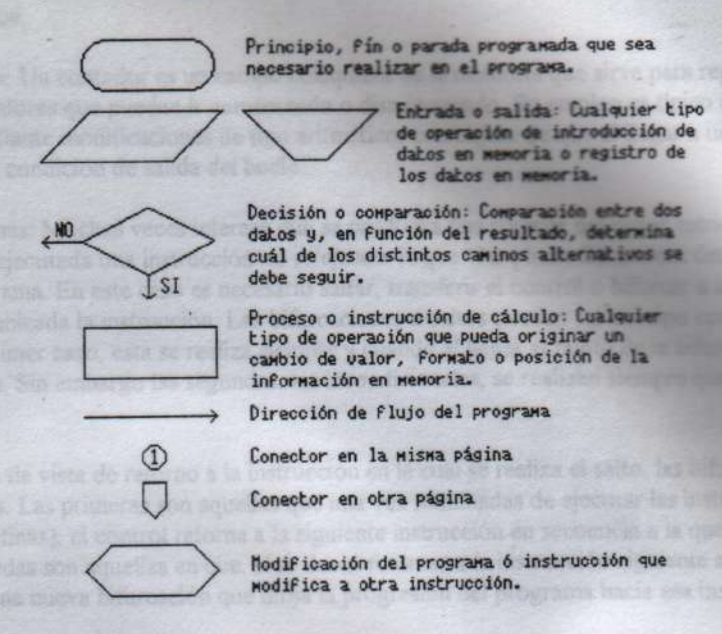


Figura 5.2 Simbolos empleados en los ordinogramas.

5.5.1. Estructuración de los ordinogramas

Siguen las mismas reglas de estructura que los programas. Siempre que sea posible, se deberán establecer las operaciones formal y estructuralmente análogas a como se van a realizar con posterioridad. Los ordinogramas pueden tener una estructura secuencial ó cíclica.

5.5.2. Principio de reciprocidad

En programas con estructura cíclica, la longitud del programa es constante, con lo que se minimiza la duración de la confección del programa y la ocupación del mismo en la memoria. En cambio la duración de la ejecución del programa es mayor.

Estas características excluyentes que ofrecen las dos estructuras que puede admitir un programa y, por tanto, su correspondiente ordinograma, constituyen el llamado principio de reciprocidad de la programación.

La diferencia fundamental estriba en que, en estructuras cíclicas, siempre es necesario programar la salida del bucle, mientras que el fin del programa secuencial es automática por su propia naturaleza.

5.5.3 Componentes de los ordinogramas

Básicamente los ordinogramas se componen de: bucles e iteraciones (repeticiones), contadores, bifurcaciones, switches y tablas en memoria interna o matrices.

1) Bucles e iteraciones: existen siempre que hay estructura cíclica. Un bucle, en general, consta de:

a) una o más instrucciones que sirven de preparación o arranque del bucle (i = 0).

b) un grupo de instrucciones que constituyen el cuerpo del bucle y que se ejecutan repetidamente (operaciones).

c) un grupo de instrucciones que modifican el bucle haciéndolo progresar (i = y+1).

d) Una instrucción de comprobación para que se pueda salir del bucle (¿es i = x?) La modificación del bucle puede realizarse de diversas formas: por operaciones aritméticas, por modificación de instrucciones por medio del registro índice.

2) Contadores: Un contador es un campo cualquiera de la memoria que sirve para representar a una variable destinada a contener valores que puedan ir aumentando o disminuyendo. Su empleo es típico dentro de un bucle, para hacerlo progresar, mediante modificaciones de tipo aritmético, con objeto de que al llegar a un valor deseado del contador, se cumpla la condición de salida del bucle.

3) Bifurcaciones: Muchas veces interesa que se produzca una ruptura del orden estrictamente secuencial, es decir, que después de ejecutada una instrucción no se ejecute la que viene inmediatamente detrás, sino otra que está en otra parte del programa. En este caso es necesario saltar, transferir el control o bifurcar a aquella parte del programa donde está ubicada la instrucción. Las bifurcaciones o saltos pueden ser del tipo condicional o incondicional. En el primer caso, esta se realiza siempre y cuando al llegar al punto de la bifurcación se satisfaga una determinada condición. Sin embargo las segundas, las incondicionales, se realizan siempre que el programa pase por el punto establecido.

Bajo el punto de vista de retorno a la instrucción en la cual se realiza el salto, las bifurcaciones pueden ser controladas o dirigidas. Las primeras son aquellas que una vez terminadas de ejecutar las instrucciones que siguieron a la bifurcación (subrutinas), el control retorna a la siguiente instrucción en secuencia a la que se produjo la bifurcación. Las segundas son aquellas en que, si se desea retornar a la instrucción siguiente a la que produjo el salto, es necesario generar una nueva bifurcación que dirija la progresión del programa hacia esa instrucción.

4) Switches: El programador, al utilizar las bifurcaciones, debe distinguir si la alteración de la secuencia de ejecución es inmediata o bien hay que demorar la decisión. En este caso se emplea la técnica del switch o conmutador. Esta técnica se basa en:

a) reserva de una posición de memoria que contenga una variable que se va a utilizar en la operación de comparación. A esta variable se le da el valor "1" (switch en ON), cuando se desea una salida en la toma de decisión diferida; y el valor "0" (switch en OFF) cuando se desea la otra salida.

b) modificación del código de operación de una instrucción que ha de ejecutarse más adelante, consistente en sustituir el código de operación de dicha instrucción por un código no operativo.

5) Tablas en memoria interna: Una de las formas más corrientes de organizar en memoria los diferentes elementos de información consiste en colocarlos en una determinada zona de memoria y unos a continuación de los otros. Una tabla establece siempre una correspondencia entre dos tipos de información: la información significativa que se pretende buscar y la información de entrada que permite encontrar aquella. Esta última se denomina con frecuencia, dirección asociativa o, simplemente, etiqueta (label). La colocación en memoria de una tabla se puede concretar en la constitución de un simple cuadro, si la posición de una información significativa en el cuadro, puede calcularse a partir de la información de entrada. Pero en el caso más general, la tabla se colocará en memoria de forma que cada elemento contenga la información de entrada y la información significativa correspondiente.

Tema 6

Conceptos de programación

6.1. Tareas de programación

Vamos a describir a continuación algunas de las tareas que podríamos realizar con el Microinstructor. Estas son:

- Cálculos de funciones simples, como suma, resta, AND lógico, etc.
- Toma de decisiones según sean los datos introducidos o los resultados de los cálculos. El sistema podrá escoger un camino u otro según sea su decisión, eso es precisamente, lo que introduce el factor inteligente.
- Lazos para repetir tareas un número concreto de veces o hasta que se satisfaga una condición.
- Manejo y proceso de grupos de datos tales como lecturas de sensores, entrada de señales de prueba, salidas de control o cadenas de caracteres.
- Conversión y manipulación de códigos, manejo de datos que son o serán codificados de alguna forma específica como BCD, ASCII, siete segmentos o Gray. Estos trabajos son esenciales para obtener datos desde, y mandarlos a, periféricos.
- Realización de funciones matemáticas tales como suma y resta en múltiple precisión, multiplicación, división y raíz cuadrada.

6.2. Registros y señales

Ya conocemos los siguientes registros:

Contador de programa (PC)
Acumulador (A)
Registros índices X e Y
Registro de estado (P)
Puntero del Stack (S)
Los flags del registro de estado son:
Negativo (N) o de signo
Overflow (V)
Comando de ruptura (B)
Modo decimal (D)
Comando de interrupción (I)
Cero (Z)
Acarreo (C)

Vamos a describir a continuación la utilización de los registros:

PC Contiene la dirección de la siguiente instrucción a ejecutar. Se incrementa automáticamente cada vez que se usa; las instrucciones de salto colocan un nuevo valor en este registro.

A Contiene el operando (y el resultado) en las operaciones lógicas y aritméticas. Es el resultado de las actividades del procesador.

X e Y Generalmente contienen contadores para la creación de lazos o índices para el manejo de tablas o grupos de datos.

S Contiene la dirección (en página 1) del valor superior depositado en el Stack, utilizado para guardar la dirección de retorno de subrutinas o los contenidos previos de registros y flags.

P Contiene los flags que representan el estado del 6502. Se describen a continuación la utilización de los flags más corrientes.

Generalmente los utilizaremos para comprobar sus valores y efectuar saltos condicionales (ver apartado 6.4).

N Se utiliza para comprobar los bits de estado, por ser el más significativo, y valores de signos.

Z Se utiliza para comprobar igualdades (después de una resta), valores nulos en un contador y valores nulos de un bit (después de un AND lógico).

C Se utiliza para guardar acarreos positivos y negativos en operaciones aritméticas, para determinar el resultado de comparaciones numéricas y para comprobar el valor de un bit (después de una operación de desplazamiento).

6.3. Operaciones simples

La mayoría de las operaciones simples se realizan en el acumulador. Estas instrucciones son:

- Suma (ADC).
- Resta (SBC),
- AND lógico (AND).
- OR lógico (ORA).
- OR EXCLUSIVO lógico (EOR).

En todas estas operaciones se supone que un operando está en el acumulador y el otro en memoria. El resultado se guarda en el acumulador. La CPU necesita dar tres pasos para realizar una operación aritmética o lógica entre dos números. Cada uno de ellos corresponde a una instrucción diferente del 6502:

1. Carga el acumulador con un operando.
2. Realizar la operación entre el acumulador y el otro operando.
3. Guardar el resultado (depositado en el acumulador) en memoria.

Ejemplo: Efectuar una suma entre los contenidos de las posiciones de memoria 500H y 501H y colocar el resultado en la posición 502H.

Mnemónico	Comentario
LDA 0500H	Carga en el acumulador el contenido de la 500H
ADC 0501H	Súmalo con el contenido de la 501H
STA 0502H	Guarda el resultado en la 503H
BRK	Fin de programa.

6.4. Toma de decisiones

En el apartado anterior el sistema actuaba como un simple calculador. Las instrucciones de salto condicional le convierten en un calculador inteligente, capaz de realizar diversas acciones según sea el dato de entrada o el resultado de una operación. La siguiente tabla muestra estas instrucciones.

Instrucción	Flag utilizado	Valor para el que ocurre el salto
BCC	CARRY (C)	0
BCS	CARRY (C)	1
BNE	CERO (Z)	0
BEQ	CERO (Z)	1
BPL	NEGATIVO (N)	0
BMI	NEGATIVO (N)	1
BVC	OVERFLOW (V)	0
BVS	OVERFLOW (V)	1

Si el Flag especificado no tiene el valor indicado, no se produce el salto.

El sistema efectúa las instrucciones de salto condicional de la siguiente forma:

Si se cumple la condición, se coloca un nuevo valor en el contador de programa (PC) y el 6502 ejecuta la instrucción almacenada en la dirección correspondiente al nuevo valor.

Si no se cumple, el contador no cambia y se sigue ejecutando las instrucciones secuencialmente.

Ejemplo: Encontrar el contenido de mayor valor de las posiciones de memoria 500H y 501H y almacenar el mayor valor en la posición 502H.

Etiqueta	Mnemónico	Comentario
	LDA 0500H	Carga en el acumulador el contenido de la 500H
	CMP 0501H	¿Es mayor el contenido de la 501H?
	BCS $ZIPI	No, salta a la dirección $ZIPI
	LDA 0501H	Si, carga en el acumulador el segundo valor ZIPI
$ZIPI	STA 0502H	Almacena el mayor valor
	BRK	Fin de programa

6.5. Bucles

La forma más sencilla de hacer repetir al microprocesador una sección, o parte de un programa, un número concreto de veces es:

1. Antes de introducir la sección a repetir, inicializar un contador a un valor específico.
2. Después de efectuar la sección, decrementar el contador.
3. Si el resultado del paso 2 no es cero, volver al comienzo de la sección.

De esta manera la instrucción de salto debe ser BNE. Se puede colocar el contador en el registro X, en el Y o en una posición de memoria. La estructura básica de un bucle es entonces (usando X):

Etiqueta	Mnemónico	Comentario
$BUCLE		\
		\ Conjunto de instrucciones
		/ a repetir
		/
	DEX	Decrementa en uno el registro X
	BNE $BUCLE	¿Es cero el registro X?. No, salta a la dirección $BUCLE
	BRK	Sí, para.

Otra alternativa sería incrementar la cuenta:

Etiqueta	Mnemónico	Comentario
$BUCLE		\
		\ Conjunto de instrucciones
		/ a repetir
		/
	INX	Incrementa el valor del registro X
	CPX #CUENTA	¿Es X = CUENTA?
	BNE $BUCLE	No, salta a la dirección BUCLE
	BRK	Sí, para.

Se necesita una instrucción más (CPX) pero este método podría ser más fácil de comprender dentro de un programa.

6.6. Agrupamiento de datos. Matrices y Tablas.

La clave para procesar grupos de datos con el microprocesador 6502 es utilizar un registro índice para fijar el número de elementos. Recordar que necesitamos dos cosas para describir un elemento en cualquier agrupamiento:

1. La dirección de comienzo.
2. El número o índice del elemento.

Podemos caracterizar un elemento en una tabla como A_i, en donde A es el nombre del grupo e i es el número del elemento. Para procesar grupos de elementos podemos usar el direccionamiento indexado. De este modo, el procesador suma el contenido de un registro índice (X o Y) a la dirección que hay en la instrucción para conseguir la dirección definitiva.

Ejemplo: Si (X) = 06H, la instrucción AND 2000H, X realiza:

(A) = (A) & (2000H + (X)) = (A) & (2006H)

Obsérvese que X e Y son registros de 8 bits, de manera que nunca podremos añadir un número mayor que 256 (FFH).

Un punto a tener en cuenta es que podemos cambiar el valor de X o Y, con INX, INY, DEX o DEY, y luego repetir la instrucción. Cada repetición obtiene el dato de una dirección diferente de la memoria.

A la dirección del dato en cada momento se le llama dirección efectiva. Podemos cambiar la dirección efectiva aunque el dato o la instrucción estén en una zona de memoria ROM.

Ejemplo: Poner a cero todas las posiciones de memoria comprendidas entre la dirección 500H y la 508H.

Etiqueta	Mnemónico	Comentario
	LDX # 00H	Poner el registro X (contador) a cero
$LAZO	STA 0500H,X	Poner a cero la posición de memoria
	INX	Incrementar el contador en una unidad
	CPX #08H	Comparar con el valor 08H
	BNE $LAZO	¿X no es igual a 08H?, saltar a la dirección $LAZO
	BRK	¿X sí es igual a 08H?, parar

Obsérvese la utilización de INX (o DEX) y CPX. Recuerde que INX y DEX sólo incrementa o decrementa el registro X en una unidad. Para incrementos mayores pueden repetirse las instrucciones o realizar la operación aritmética necesaria en el acumulador. No existen instrucciones para realizar operaciones matemáticas con los registros X e Y.

6.7. Conversión y manipulación de códigos.

El microprocesador 6502 se puede utilizar para manejar datos en cualquier código, aunque por sí mismo sólo maneja valores numéricos. No importa si el dato va a ir a un periférico o va a un operador. Es decir, el procesador maneja caracteres de control, letras dígitos, etc. todos de la misma forma. El programador debe proporcionar la inteligencia en forma de programa para que distinga los valores adecuadamente.

Ejemplo: Colocar el contenido de la posición 501H a 1 si la posición 500H contiene un numero decimal codificado en ASCII, o a cero, si no es así. Recordar que el código ASCII representa a los números decimales como 30H, 31H, ...,39H en hexadecimal.

Etiqueta	Mnemónico	Comentario
	LDA 0500H	Obtener el número de la posición 500H
	CMP #30H	¿Está por debajo del cero en ASCII?
	BCC $PEPE	Sí, entonces no vale, salta a la posición $PEPE
	CMP #3AH	¿Está por encima del nueve en ASCII?
	BCS $PEPE	Sí, no vale, salta a la posición $PEPE
	INX	Válido si está entre 0 y 9. Pon a 1 el registro X
PEPE	STX 0501H	Coloca el valor de X en la posición 501H
	BRK	Para.

Algunas conversiones de código, como de ASCII a decimal o a la inversa, se pueden realizar como simples operaciones aritméticas. Las conversiones más complejas, como por ejemplo de ASCII a siete segmentos, se pueden manejar por medio de tablas. Estas tablas son agrupaciones de datos que están almacenadas en memoria en un orden adecuado.

Por ejemplo, el elemento n-ésimo de un grupo puede contener simplemente el código que corresponde al elemento o caracter n. Si la tabla comienza, por ejemplo en la dirección $TABLA. El programa para convertir un elemento del acumulador en su equivalente será:

Etiqueta	Mnemónico	Comentario
	TAX	Haz el registro X = A
	LDA $TABLA, X	Carga en el acumulador el nuevo valor del elemento.

6.8. Aritmética.

Las únicas instrucciones aritméticas disponibles en el 6502 son:

- ADC; Suma con acarreo, (A) = (A) + (M) + C
- SBC; Resta con acarreo, (A) = (A) - (M) - 1 + C

Las operaciones aritméticas más complejas, como la multiplicación, la división o la raiz cuadrada deben ser programadas.

Deben observarse los siguientes aspectos:

1. El bit de CARRY siempre se incluye. Si no se desea que su valor afecte a la operación, debe ponerse a cero con la instrucción CLC antes de efectuar una suma o ponerlo a uno con la instrucción SEC antes de una resta.
2. Todas las sumas y restas son decimales (BCD) si D = 1. Pero no olvidar poner D a cero (CLD) cuando se haya terminado de usar la aritmética decimal.
3. Se puede usar ASL A para sumar el acumulador consigo mismo.
4. Se puede utilizar ADC #00H o SBC #00H para sumar C al acumulador o restar 1-C del acumulador.
5. ADC y SBC son ambas operaciones con 8 bits en las que el resultado final se deposita en el acumulador.
6. El bit V nos dice si ha ocurrido un desbordamiento en complemento a dos, es decir, si la magnitud del resultado ha afectado a su bit de signo.

Veamos algún ejemplo simple:
Ejemplo 1: Sumar el contenido de las posiciones 500H y 501H y colocar el resultado en la 502H.

Etiqueta	Mnemónico	Comentario
	CLC	Pon el Carry a cero
	LDA 0500H	Coge el primer operando
	ADC 0501H	Suma el acumulador con el segundo operando
	STA 0502H	Almacena la suma en la 502H
	BRK	

Ejemplo 2: Sumar un número de 16 bits depositado en las posiciones de memoria 500H y 501H (este último es el más significativo) al número de 16 bits colocado en 502H y 503H (este último es el más significativo) y colocar el resultado en 504H y 505H (el más significativo en la 505H).

Etiqueta	Mnemónico	Comentario
	CLC	
	LDA 0500H	
	ADC 0502H	Suma los 8 bits menos significativos
	STA 0504H	
	LDA 0501H	
	ADC 0503H	Suma los 8 bits más significativos
	STA 0505H	
	BRK	

En este último ejemplo debemos observar que necesitaremos el acarreo producido por la suma de los 8 bits menos significativos, si es que hubo.

6.9. Subrutinas.

El modo más simple de acceder a una rutina desde cualquier punto de otros programas es darle tratamiento de una subrutina. Esto requiere:

1. Una instrucción de salto a la subrutina (JSR) en el programa principal, para darle el control.
2. Una instrucción de retorno (RTS) al final de la subrutina para devolver el control al programa principal.

Nota: Sólo se necesita tener una copia en memoria de la subrutina. El control se devuelve automáticamente a la instrucción siguiente a JSR, una vez que se ejecutó la subrutina y se recibió la orden de retorno.

¿Cómo sabe el procesador dónde tiene que volver?. La respuesta está en la utilización de un STACK en memoria RAM y el puntero de Stack. Las dos instrucciones de subrutina, JSR y RTS, trabajan de esa forma. La instrucción JSR almacena los 8 bits más significativos del contador de programa (PC) en la dirección marcada por el puntero del Stack (S), decrementa esta dirección y almacena en ella los 8 bits menos significativos del contador y vuelve a decrementar el puntero.

Se puede simbolizar lo que ocurre de la siguiente manera:

($0100 + (S)) = (PC)Alto
(S) = (S) - 1
($0100 + (S)) = (PC)Bajo
(S) = (S) - 1

La instrucción RTS realiza la función opuesta. Incrementa el puntero y trae el contador de programa desde esas posiciones:

(S) = (S) + 1
(PC)Bajo = ($0100 + (S))
(S) = (S) + 1
(PC)Alto = ($0100 + (S))

Ambas instrucciones tienen las siguientes características:

1. El Stack está siempre en página 1. Esto es, los 8 bits más significativos de su dirección valen siempre 01H, mientras que los 8 menos significativos los da el contenido del registro S.

2. El Stack se mueve hacia posiciones más bajas de memoria, desde las direcciones más altas. Obsérvese que el área ocupada por el Stack es una zona normal de lectura/escritura; lo único que cambia es el valor en el puntero del Stack.
3. El programa completo debe inicializar y manejar el Stack.
4. Las subrutinas pueden llamar a su vez a otras subrutinas, puesto que las direcciones de retorno se guardan en el Stack. Recordar que la última dirección depositada es la primera en salir; así mismo la primera dirección depositada será al última en salir.
5. Los contenidos de los registros se pueden guardar también en el Stack, utilizando las instrucciones PHA y PHP (agregando TXA y TYA para guardar los registros X e Y) y recogerlos del Stack utilizando PLA y PLP (agregando TAX o TAY para que pasen a los registros X e Y).

6.10. Desarrollo de los programas.

Desde luego, el desarrollo de los programas requiere algo más que la escritura de cortas secuencias de instrucciones. El diseñador de software debe también:

- Definir el problema.
- Diseñar el programa.
- Depurar, comprobar y documentar el programa.

Estas actividades normalmente toman más tiempo que el de la escritura de las instrucciones. Una regla aproximada es que el reparto del tiempo puede ser:

- 40% para la definición y el diseño.
- 20% para la escritura de las instrucciones (codificado).
- 40% para el depurado, comprobación y documentación.

Naturalmente no es el momento ni el lugar para justificar cualquiera de estas afirmaciones. Si alguien desea más detalles, u ocuparse de aspectos más concretos sobre la programación, le sugerimos que consulte algunas de las obras sobre programación que existen en la biblioteca del Centro o en el Departamento.

Tema 7 Utilización del MI-650C

A continuación se describen algunas de las funciones que posee el MI-650C (hay que tener en cuenta que varias de ellas no son descritas por pensar que son de carácter secundario para el interés de estos apuntes).

El aspecto que tiene el teclado del Microinstructor es el mostrado en la figura 7.1.

Fig. 7.1. Teclado del MI-650C

Donde se pueden ver cómo varias teclas tienen una doble función: el valor hexadecimal asignado a cada una de ellas y una función que facilita la operatividad del Microinstructor.

El Microinstructor indica en cada paso de introducción de órdenes qué tipo de dato está esperando, e interpreta la tecla pulsada como función o como valor numérico según corresponda. De esta forma tras la conexión del equipo y en el estado de reposo, el Microinstructor espera una función que interpretará como la primera tecla pulsada. A partir de este momento las teclas pulsadas serán interpretadas como datos hasta la finalización de la instrucción, tras pulsar la tecla [FIN].

Cuando el Microinstructor espera datos, enciende los puntos para indicar el lugar y tamaño del dato solicitado. Las direcciones son introducidas en los cuatro displays de la izquierda y el contenido de éstas es introducido en los dos displays de la derecha.

El estado de reposo del Microinstructor se establece tras pulsar las teclas [RESET] o [BREAK] y se reconoce por el encendido de los segmentos centrales de todos los Displays.

7.1. Inicialización y puesta a cero de registros

La inicialización y puesta a cero de registros se consigue pulsando la tecla [RESET]. Tras pulsar esta tecla se produce el mismo efecto que al conectar el Microinstructor: se abandona la ejecución del programa de usuario (en caso de estar realizándose), se inicializan los registros y se efectúa un test de los diodos LED y microinterruptores de la VIA de usuario, pasando finalmente al estado de reposo.

Al pulsar la tecla [STOP/BRK] se abandona la ejecución del programa de usuario al igual que en el caso anterior, pero no se inicializan los registros y, finalmente, se pasa al estado de reposo.

La tecla [STOP/BRK] se utiliza como ayuda en la depuración, para abandonar el programa de usuario en curso, cuando éste está perdido en un punto incontrolado, por ejemplo en un bucle sin fin; al no ser inicializados los registros podemos consultar los valores que tenían en el momento de producirse la rotura del programa.

La tecla [RESET] se utiliza para la puesta a cero de registros y para retomar el control en aquellos casos en que el Microinstructor se encuentre incontrolado.

RESET — INICIALIZA EL MICROINSTRUCTOR

STOP BRK — ABANDONO DE PROGRAMAS SIN INICIALIZACIÓN

7.2. Visualización y modificación de la memoria

7.2.1. Visualización de la memoria

Tras encender el Microinstructor y estar en el estado de reposo se deben seguir los siguientes pasos para visualizar el contenido de una posición de memoria.

1° Pulsar [MEM/0]. En el display aparece:

En el campo de direcciones se encienden los puntos y aparece una dirección; esta dirección puede ser cambiada.
En el campo de datos aparece el texto "ad" indicando que el número en el campo de dirección es una dirección de memoria.

2° Pulsar [0] [4] [0] [0]. Cada vez que pulsamos una tecla, el valor asignado a ésta se coloca en el display de la derecha del campo de direcciones y se corren hacia la izquierda los valores anteriores, desapareciendo el de la izquierda. De esta forma se pueden teclear tantos dígitos como se desee, quedando válido el número presente en el display.

3° Pulsar [FIN]. Esta función indica la finalización de la introducción de la dirección a la que deseamos acceder. Al mismo tiempo, desaparece el mensaje "ad" y aparece el contenido de la posición de memoria tecleada anteriormente. Se encienden los puntos del campo de datos, indicando que puede ser cambiado el contenido de la dirección mostrada en el campo de dirección.
Pulsando la tecla [+], o la tecla [-] se avanza o retrocede una posición en la dirección que se está visualizando.
Pulsando la tela [FIN] termina la función de visualización, pasando el Microinstructor al estado de reposo.

MEM 0 — VISUALIZACIÓN DE LA MEMORIA

MEM 0 — A.D. PIDE DIRECCIÓN

FIN — DATO CONTENIDO EN LA DIRECCIÓN

\+ — DIRECCIÓN SIGUIENTE

\- — DIRECCIÓN ANTERIOR

7.2.2. Modificación de la memoria

Para modificar la memoria hay que tener en cuenta que debemos estar en memoria RAM, en caso contrario no podremos modificarla.

La dirección 0400H pertenece a memoria RAM por lo que podremos modificar su contenido.

Para introducir el dato "EA" en la posición 0400H, los pasos 1°, 2° y 3° del punto anterior son los mismos y, una vez realizados, se pulsará el valor hexadecimal que se desea introducir, o sea, se pulsa [E] [A].

En este momento el dato no se ha almacenado en la memoria, para que lo haga hay que pulsar la tecla [+] o la tecla [-] avanzando o retrocediendo al mismo tiempo una posición en las direcciones de memoria. La secuencia de teclas que hay que pulsar para conseguir el efecto deseado será pues:

1ª [MEM/0] [0] [4] [0] [0] [FIN] [E] [A] [+], o bien
2ª [MEM/0] [0] [4] [0] [0] [FIN] [E] [A] [-]

La comprobación de que está correctamente modificada la posición de memoria deseada se puede hacer pulsando la tecla [-] o [+] para volver a la dirección modificada (para el 1er caso el [-] y para el 2º el [+]).

Si se desean modificar varias posiciones de memoria seguidas basta con colocarse en la primera de ellas y modificarla pulsando como última tecla el [+], con ello además se consigue avanzar una posición de memoria que, a su vez, puede ser modificada.

Como en el caso anterior, pulsando la tecla [FIN] termina la función de modificación, pasando el Microinstructor al estado de reposo.

Si se intenta modificar el contenido de una posición de memoria ROM, el Microinstructor no lo permite y, tras pulsar la tecla [+] o [-], no avanza o retrocede la dirección y no es introducido en memoria el dato escrito en el campo de datos. Esto se puede comprobar al intentar escribir el dato EA en la dirección E000H.

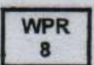
7.3. Protección de la RAM contra escritura

Esta función afecta a la RAM comprendida entre las direcciones 0400H y 1FFFH. La tecla que realiza esta función es [WPR/8].

1º Estando en el estado de reposo del Microinstructor se comprueba que puede ser modificado el contenido de la RAM.
2º Pulsamos la tecla [WPR/8]. Aparece el mensaje "on", indicando que la memoria ha sido protegida y pasando al estado de reposo.
3º Estando de nuevo en el estado de reposo del Microinstructor se comprueba cómo la RAM no puede ser modificada como si fuese memoria ROM.

Para eliminar la protección de la RAM pulsar de nuevo la tecla [WPR/8]. Aparece el mensaje "oFF", indicando que la memoria RAM ha sido desprotegida y ya puede volver a ser modificada.

Nota:
Cuando pulsamos la tecla [RESET], siempre deja la RAM sin protección, de manera que se pueden escribir sobre ella los programas que se deseen.



PROTECCIÓN CONTRA ESCRITURA ON/OFF

7.4. Visualización de los registros internos de la CPU

La tecla que realiza esta función es [REG/2]. Lo que en realidad nos muestra es el valor contenido en las posiciones de memoria que hacen de imagen de los registros internos de la CPU (también podrían ser consultadas estas posiciones y veríamos el mismo valor).

1º Estando en el estado de reposo pulsamos la tecla [REG/2]. En el campo de direcciones del display, nos muestra el dato que contiene el registro, indicado en el campo de datos. Así en el campo de direcciones indicará "F.7d0." y en el de datos "Pc".
2º Pulsando la tecla [+] o [-] se pasa al registro siguiente o anterior, respectivamente, visualizando así todos los registros.
3º Con la tecla [FIN] finaliza la visualización y se pasa al estado de reposo.

Nota:
En el campo de direcciones aparecen encendidos los puntos indicando que pueden ser cambiados estos valores. Estos cambios serán efectivos sólo si, tras ellos, es pulsada la tecla [+] o [-].

El orden de visualización de los registros si pulsamos la tecla [+], es el siguiente (si la tecla pulsada es la [-] será el orden inverso).

Contenido	Registro	
F.7.D.0.	PC	Contador de Programa.
F.F.	S	Puntero de Stack.
0.0.	AC	Acumulador.
0.0.	X	Registro X.
0.0.	Y	Registro Y
0.0.	P	Registro de Flags o Status.
F.7.D.0.	PC	Contador de Programa.

El contenido de estos registros sólo será el indicado arriba en el caso de haber sido pulsada la tecla [RESET] o ser encendido el Microinstructor.

REG 2 — VISUALIZACIÓN DE LOS REGISTROS

REG 2 — P C. MUESTRA DE CONTADOR DE PROGRAMA

\+ — SIGUIENTE REGISTRO

\- — ANTERIOR REGISTRO

7.5. Ejecución de programas

La ejecución de programas puede ser tanto de programas escritos por el usuario en la memoria RAM como de programas residentes en memoria EPROM. Se puede hacer de dos formas: continuado o paso a paso. El segundo método es utilizado para ayudar en la depuración de los programas de usuario que no funcionan correctamente.

Para poder ejecutar un programa es necesario que éste exista, para ello ejecutaremos el programa de inicialización del monitor. Este programa se encuentra a partir de la dirección FF71H, y está en ejecución continuamente.

7.5.1. Ejecución continuada

La ejecución en modo continuo hace que se ejecute el programa en tiempo real con el Microinstructor pierda el control mientras se está ejecutando. Éste sólo recupera el que trol si termina el programa con un RTS (retorno de subrutina), o si se aborta el programa pulsando las teclas [STOP/BRK] o [RESET] y, por supuesto, si se desconecta y vuelve a conectar el Microinstructor. La tecla encargada de realizar esta función es [GO/1].

Para realizar esta ejecución se deben seguir los pasos siguientes:

1° Pulsar [GO/1].
2° A continuación, la dirección de comienzo del programa que se quiere ejecutar [F] [F] [7] [1].
3° Pulsar [FIN]. Ahora el programa se está ejecutando y realiza las mismas operaciones a la conexión del Microinstructor.

Nota:

En aquellos programas en los que no se realiza un control de los displays, éstos estarán apagados mientras el programa se está ejecutando, y pasarán a encender el segmento central cuando termine el programa y el Microinstructor pase al estado de reposo.

[GO / 1] EJECUCIÓN CONTINUADA

[GO / 1] A.D. PIDE DIRECCIÓN DE EJECUCIÓN

[FIN] PROGRAMA DE EJECUCIÓN

7.5.2. Ejecución paso a paso

Este tipo de ejecución es utilizada para depurar programas de usuario y para seguir los valores que toman los registros de la CPU.

La activación o desactivación de la Ejecución Paso a Paso, se consigue pulsando la tecla [RUN/STEP] y su estado lo indica un diodo LED amarillo, de tal forma que si se encuentra encendido indica Ejecución Paso a Paso y si se encuentra apagado indica Ejecución Continuada.

Para realizar esta ejecución se deben seguir los pasos siguientes:

1° Con objeto de realizar la ejecución debe ser introducido en memoria un programa; éste puede ser el siguiente: a partir de la dirección de RAM 0400H se introducen los bytes, "A5 00 85 04 60".

2° Pulsar [RUN/STEP].

3° Pulsar [GO/1].

4° A continuación, la dirección de comienzo del programa que se quiere ejecutar [0] [4] [0] [0].

5° Pulsar [FIN]. En el display aparece "0402" en el campo de direcciones y "85" en el campo de datos. Lo que se ha hecho es ejecutar la primera instrucción y está indicando cuál es la dirección de la siguiente que se va a ejecutar y el contenido de dicha posición. En este punto puede ser visualizado el contenido de los registros de la CPU y así comprobar si se ha ejecutado correctamente la instrucción.

6° Pulsar [STEP/C]. Se ha ejecutado otra instrucción y de nuevo aparece en el display, en el campo de dirección, la dirección de la siguiente instrucción por ejecutar, "0404", y en el campo de datos el contenido de esta instrucción "60". Cada vez que se pulse la tecla [STEP/C] será ejecutada una instrucción del programa y se indicará cuál es la siguiente.

Nota:

Cuando se intenta ejecutara un programa paso a paso y éste se encuentra en EPROM, la ejecución se realiza continuada ignorando el estado de la tecla [STEP/C].

[RUN / STEP] EJECUCIÓN PASO A PASO

[GO / 1] EJECUCIÓN

A.D. PIDE DIRECCIÓN DE EJECUCIÓN

[FIN] INSTRUCCIÓN EJECUTADA

[STEP / C] EJECUCIÓN INSTRUCCIÓN SIGUIENTE

MUESTRA DIRECCIÓN SIGUIENTE, Y CONTENIDO, A LA EJECUTADA

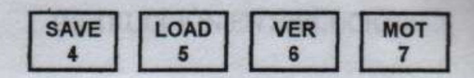
7.6. Utilización del cassette

El sistema está dotado de una salida de datos, preparada para ser transferidos hasta un cassette mono con control remoto, y del correspondiente programa de control que gobierna esta transmisión.

El cassette es utilizado para almacenar programas de un tamaño considerable o programas que son utilizados con mucha frecuencia. Su utilización se hace indispensable como lugar de almacenamiento de programas, dado su bajo coste y fácil empleo.

El cassette y el Microinstructor deben ser conectados por un cable; en caso de no poseerlo, se puede fabricar fácilmente como se indica en el manual del Microinstructor.

Las teclas utilizadas para el control del cassette son:



7.6.1. Control del motor

La tecla encargada de activar o desactivar el motor del cassette es [MOT/7]. Cuando se pulsa aparece el mensaje "on", indicando que se puede trabajar con el cassette, y cuando se pulsa de nuevo aparece el mensaje "oFF", indicando que el motor del cassette está inoperante.



CONTROLA MOTOR ON, OFF

7.6.2. Salvar datos en cassette

El Microinstructor tiene la capacidad de almacenar un bloque de memoria en una cinta de cassette, independientemente de que este bloque sea de memoria RAM o ROM. Para almacenar un bloque hay que hacerlo con un número de identificación que se utilizará para su posterior recuperación.

Antes de pasar a salvar, asegurarse de que:

- El cassette se ha conectado al Microinstructor, que está alimentado, y en su interior la cinta ha sido colocada correctamente.
- Los controles de volumen y tono están en la posición correcta.
- En caso de que exista control remoto del motor, estén pulsadas las teclas de PLAY y REC. En caso de no disponer del control remoto del motor, antes de pulsar la tecla [FIN] del punto 4º pulsar las teclas del cassette PLAY y REC simultáneamente y esperar 3 ó 4 segundos antes de pulsar la tecla [FIN].
- Es conveniente que el cassette disponga de cuenta vueltas y tomar nota de la vuelta actual de la cinta y de la vuelta final donde termina el bloque grabado.

Los pasos que se seguirán para salvar el bloque de memoria en el cassette son los siguientes:

1º Pulsar la tecla [SAVE/4]. El Microinstructor solicitará el número de identificación de dicho bloque escribiendo "Id", este puede ser cualquier número hexadecimal menos el 00H y FFH que tienen una función especial en la recuperación de datos.

2º Introducir el número de identificación y pulsar [FIN].

3º El Microinstructor solicitará la dirección inicial del bloque de datos por grabar con el indicador "dI". Se introduce esta dirección y se pulsa [FIN].

4º El Microinstructor solicitará la dirección final del bloque de datos con el indicador "dF". Se introduce la dirección y al pulsar [FIN] comienza la transmisión de datos.

Durante la grabación, los displays permanecerán apagados y, al finalizar ésta, pasará el Microinstructor al estado de reposo.

SAVE 4	CARGA PROGRAMA DE MICRO A CASSETTE	
SAVE 4		I.D. PIDE IDENTIFICACIÓN
FIN		D.I. DIRECCIÓN INICIAL
FIN		D.F. DIRECCIÓN FINAL
FIN	INSTRUCCIÓN EN EJECUCIÓN	

7.6.3. Verificación de los datos salvados en cassette

Es conveniente después de grabar un bloque de memoria en el cassette verificar que se ha hecho correctamente. Esta verificación se realiza en el Microinstructor Byte a Byte.

Los pasos para la verificación son los siguientes:

1° Rebobinar la cinta hasta una vuelta anterior a la de inicio de la grabación del bloque de memoria.
2° Detener el cassette y colocarlo en reproducción si dispone de control remoto.
3° Pulsar la tecla [VER/6]. El Microinstructor solicitará el número de identificación de los datos por verificar.
4° Introducir el mismo número de identificación con el que se grabó el bloque de memoria y pulsar [FIN].
5° El display presentará una "S", indicando que se esperan datos. Si hay control remoto el cassette se pondrá en marcha; en caso contrario, apretar la tecla PLAY del cassette.

Cuando el programa encuentre datos desaparecerá la "S", y si el identificador corresponde al número solicitado comenzará la verificación de los datos grabados, comparándolos con los de memoria pero sin alterarla.

Si todo ha sido correcto el Microinstructor permanecerá en el estado de reposo, en caso contrario mostrará uno de los errores:

- Err 01: Los datos de la cinta no pueden ser leídos correctamente.
- Err 02: Existe error en los caracteres de control calculados por el programa (checksum).
- Err 03: Los datos leídos no son iguales a los existentes en memoria.

En cualquier caso debe ser repetido el proceso de grabación y verificación.

Cuando el error producido es el Err 03, pulsando la tecla [MEM/0], aparece en el display la dirección de memoria que contiene el dato que no coincide.

VER 6	VERIFICAR PROGRAMA GRABADO EN CASSETTE CON UNO DEL MICRO	
VER 6		I.D. PIDE IDENTIFICACIÓN
VER 6	INSTRUCCIÓN EN EJECUCIÓN	

7.6.4. Recuperación de los datos del cassette

La recuperación de los datos del cassette sólo se puede hacer sobre la memoria RAM.
Los pasos para recuperar la información del cassette sobre la memoria RAM son los siguientes:

1° Colocar la cinta una vuelta anterior a la de inicio de la grabación del bloque de memoria y comprobar que el cassette está conectado y los mandos de volumen y tono ajustados.

2° Colocar el cassette en reproducción si dispone de control remoto.

3° Pulsar la tecla [LOAD/5]. El Microinstructor solicitará el número de identificación del programa. Se pueden dar tres casos:

- Introducir el número correspondiente al bloque de datos por recuperar. Será recuperado el bloque con dicho número de identificación.
- Introducir como identificador el 00H. Será recuperado el primer bloque de datos encontrado, con independencia de su número de identificación.
- Introducir como identificador el FFH. Será recuperado el primer bloque de datos encontrado pero colocándolo en otra zona de memoria diferente a la que tenía en la grabación.

4° En caso de emplear el identificador FFH, se deberá responder con las direcciones inicial y final del lugar de memoria donde ha de ir el programa grabado en cinta. Así, si al grabar la cinta se dieron las direcciones 0400H y 040FH como inicio y final, y se desea recuperar a partir de la dirección 0500H, debe responderse con 0500H y 050FH a las direcciones inicial y final solicitadas.

5° Pulsar [FIN]. El cassette se pondrá en marcha; en caso de no tener control remoto pulsar PLAY. Aparecerá la letra "S" hasta que se encuentren datos en la cinta con el identificador indicado. Mientras son introducidos los datos, el primer display se enciende sin que ello tenga importancia y, al mismo tiempo, se enciende un LED verde indicando la carga de datos.

Si todo va bien, al finalizar la operación, el Microinstructor pasará al estado de reposo; en caso contrario puede darse uno de estos errores:

- Err 01: Los datos de la cinta no pueden ser leídos correctamente.
- Err 02: Existe error en los caracteres de control calculados por el programa (checksum).
- Err 03: No pueden ser tomados los datos, bien porque se intenta almacenar sobre memoria ROM, o porque la RAM está protegida contra escritura.

En caso de error, revisar las posiciones del volumen y tono del cassette, inspeccionar la cinta y los cabezales y revisar la correcta conexión del cassette y el Microinstructor.

[LOAD 5] CARGA PROGRAMA DE CASSETTE AL MICRO

[LOAD 5] [| | . | .] [I | d]

[FIN] INSTRUCCIÓN EN EJECUCIÓN

IDENTIFICACIÓN DOS NÚMEROS O LETRAS
* 00 CARGA PRIMER PROGRAMA QUE ENCUENTRE
* FF TAMBIÉN CARGA EL PRIMER PROGRAMA PERO PODEMOS CAMBIAR SU POSICIÓN EN LA MEMORIA

S DESAPARECE AL CABO DE 6 SEG Y EN SU LUGAR APARECEN UNA SERIE DE SEGMENTOS

EJERCICIOS PROPUESTOS

1° Visualiza el contenido de la dirección 0400H a la 0405H, y de la dirección FFF0H a la FFFFH.
2° Modifica el contenido de la dirección 0400H a la 040FH con el dato FFH.
3° Idem al ejercicio 2°, pero de la dirección FFF0H a la FFFFH. ¿Qué diferencias observas?
4° Modifica el contenido de la dirección A400H. Observa cómo se iluminan los LED.
5° Protege la RAM contra escritura e intenta repetir el ejercicio 2° con el dato 00H. ¿Qué ocurre ahora?
6° Guarda el bloque de memoria comprendido entre las direcciones FF00H y FFFFH en el cassette. Verifícalo e intenta más tarde volver a recuperar los datos en esas posiciones. ¿Qué ocurre? ¿Por qué?
7° Intenta ahora recuperar los datos sobre las direcciones de la 0400H a la 04FFH. Comprueba que se han recuperado correctamente.

Tema 8 Elementos de entrada y salida del Microinstructor

8.1. VIA 6522 (Versatile Interface Adapter)

La VIA 6522 constituye el ejemplo típico de un adaptador universal de periféricos. Está totalmente libre para prácticas y disponible a través del *conector de aplicación* del Microinstructor.

Está constituido por los elementos principales siguientes:

- Dos Puertos (Port A y B) de 8 líneas cada uno. Cada Puerto se puede configurar como entrada o como salida de forma individual.
- Cuatro líneas de control y estado, dos para cada puerta. Se denominan CA1, CA2, CB1 y CB2.
- Un registro de desplazamiento (SR) de 8 bits, encargado de la conversión de la información serie a paralelo y viceversa.
- Dos contadores/temporizadores (Timer) de 16 bits, que pueden usarse para contar o generar pulsos.
- Lógica de interrupción, en la que se incluye un registro de flags o banderas, señalizadores de interrupción, cuyos bits indican la producción de una determinada interrupción, según su estado. También dispone de la lógica necesaria para permitir o no las interrupciones.

En la figura 8.1 aparece detallado el diagrama en bloques de la VIA 6522.

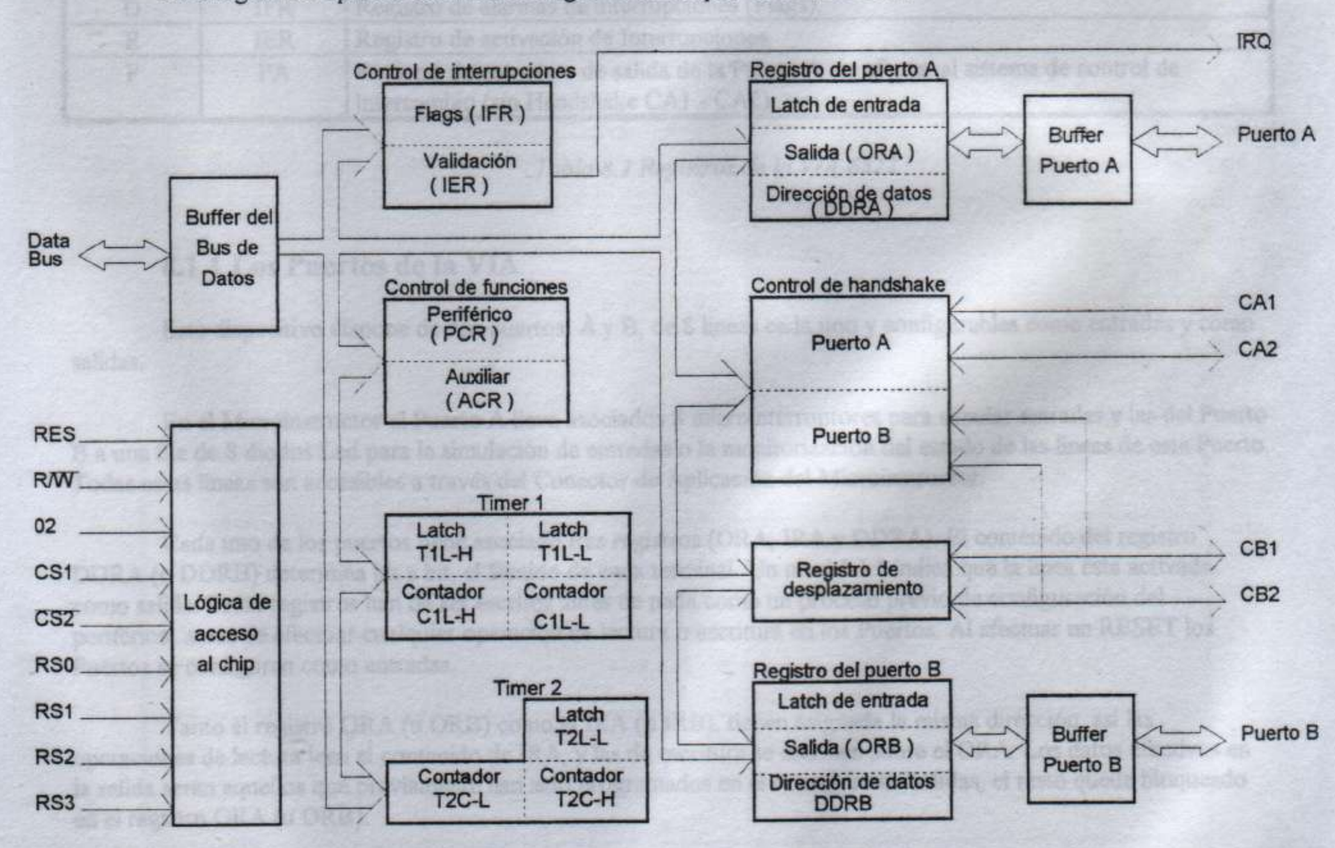


Fig. 8.1 Diagrama de bloques de la VIA 6522.

Entre las señales más importantes de acceso a la VIA se pueden destacar las cuatro entradas de selección de registros, que permiten el acceso a uno de los 16 registros internos que posee. Estas señales van conectadas a las líneas de menos peso del Bus de Direcciones. Las direcciones de acceso de los registros internos de la VIA quedan comprendidas entre XXX0H y XXXFH. Los bits de mayor peso de esta dirección vienen definidos por el circuito decodificador formado por IC1-IC7. La VIA de entrada/salida de usuario utilizada en el Microinstructor se selecciona con la señal /SEL0 (A400H-A4FFH).

En la tabla 8.2 podemos observar como seleccionar uno de los registros internos de la VIA.

Dirección relativa	Abreviatura	REGISTRO ACCEDIDO
0	PB	Registro de entrada o salida de la Puerta B.
1	PA	Registro de entrada o salida de la Puerta A, afecta al sistema de control de intercambio (con handshake automático) (CA1 - CA2).
2	DPB	Configuración Puerta B; "0" entrada, "1" salida.
3	DPA	Configuración Puerta A; "0" entrada, "1" salida.
4	T1L-L	Se carga el registro del contador T1, byte bajo; se lee el estado del contador T1, byte bajo.
5	T1L-H	Se carga el registro del contador T1, byte alto; se repone la interrupción y se transfiere todo el registro al contador; empieza a contar, se lee el estado del contador T1, byte alto.
6	T1L	Registro del contador T1, byte bajo.
7	T1H	Registro del contador T1, byte alto. Al escribir se repone la interrupción.
8	T2L	Se carga el registro del contador T2, byte bajo; se lee el estado del contador T2, byte alto y se repone la interrupción.
9	T2H	Se carga el registro del contador T2, byte alto; se repone la interrupción y se transfiere el registro al contador, empezando a contar. Se lee el estado del contador T2, byte alto.
A	SR	Registro de Desplazamiento (actúa como memoria en el modo "000"). Al leer o al escribir se repone la interrupción.
B	ACR	Registro Auxiliar de Control (Temporizador, registro de desplazamiento, etc.)
C	PCR	Registro de Control de Periférico (CA1, CA2, CB1 y CB2).
D	IFR	Registro de alarmas de interrupciones (Flags).
E	IER	Registro de activación de Interrupciones.
F	PA	Registro de entrada o de salida de la Puerta A sin afectar al sistema de control de intercambio (sin Handshake CA1 - CA2).

Tabla 8.2 Registros de la VIA 6522

8.1.1 Los Puertos de la VIA

Este dispositivo dispone de dos puertos, A y B, de 8 líneas cada uno y configurables como entradas y como salidas.

En el Microinstructor el Puerto A lleva asociados 8 microinterruptores para simular entradas y las del Puerto B a una fila de 8 diodos Led para la simulación de entradas o la monitorización del estado de las líneas de este Puerto. Todas estas líneas son accesibles a través del Conector de Aplicación del Microinstructor.

Cada uno de los puertos tiene asociado tres registros (ORA, IRA y DDRA). El contenido del registro DDRA (o DDRB) determina bit a bit, el sentido de cada terminal. Un nivel " 1 " indica que la línea esta activada como salida. Estos registros han de ser escritos antes de nada como un proceso previo de configuración del periférico, antes de efectuar cualquier operación de lectura o escritura en los Puertos. Al efectuar un RESET los Puertos se configuran como entradas.

Tanto el registro ORA (u ORB) como el IRA (o IRB), tienen asignada la misma dirección, así las operaciones de lectura leen el contenido de IRA, y las de escritura se escriben sobre el ORA. Los datos efectivos en la salida serán aquellos que previamente han sido programados en el Puerto como salidas, el resto queda bloqueado en el registro ORA (u ORB).

Al efectuar una operación de lectura del Puerto, se obtiene el dato correspondiente del registro IRA (o IRB). Este registro tienen dos modos de funcionamiento:

1. **Modo transparente***:* Su contenido es el actual de las líneas del Puerto.
2. **Modo Latch:** Su contenido memoriza el estado de las líneas del Puerto en el instante en que una señal auxiliar de control es activada.

En cualquier momento se pueden leer las líneas del Puerto aunque este configurado como salida. En este caso, la lectura refleja el estado real de las líneas del Puerto.

El registro ACR (Registro auxiliar de control), controla estos dos modos de funcionamiento mediante los bits 0 y 1. En modo Latch la orden de captura se recibe del exterior a través de las líneas auxiliares CA1 o CB1.

8.1.2 Líneas auxiliares de control

Estas líneas pueden tener distintas funciones:

CA1 y CB1: Entrada de interrupción.
Entrada de orden de memorización, en modo Latch.
Entrada de control para transferencia de datos.

CA2 y CB2: Entrada de interrupción.
Salida de control, para transferencia de datos.
Salida de impulsos.
Salida programable.

La adecuada programación del registro PCR determina cada uno de los modos de funcionamiento de estas líneas.

8.1.3 Otros registros de la VIA

En el apartado anterior se han visto los registros DDRA y DDRB, a continuación procederemos a estudiar otros registros fundamentales de la VIA. Estos son:

Registro de Control de Periféricos (PCR)

Determina la polaridad de las tensiones activas sobre las líneas de entrada de estado o de control CA1, CA2, CB1 y CB2, se encuentra en la dirección relativa de memoria A40CH. Su configuración queda reflejada en la figura y tabla 8.3.

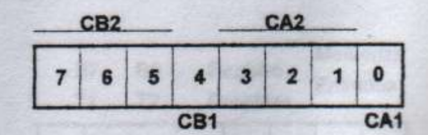


Fig. 8.3 Registro de control (PCR)

CA1	Bit 0	0 => Línea activa flanco descendente 1 => Línea activa flanco ascendente		
CA2	Bit 1	Entrada B3 = 0	0 => Reposición automática de interrupción al leer o escribir en Puerto A 1 => Reposición manual de bit cero en registro de alarmas	
		Salida B3 = 1	Automático B2 = 0	0 => CA2 sube con CA1 y baja al leer la puerta A 1 => CA2 baja un pulso de un ciclo al leer la puerta A
			Manual B2 = 1	0 => CA2 fijo en bajo 1 => CA2 fijo en alto
	Bit 2	Entrada B3 = 0	0 => Línea CA2 activa al flanco descendente 1 => Línea CA2 activa al flanco ascendente	
		Salida B3 = 1	0 => Salida CA2 Automática (Handshake) 1 => Salida CA2 manual	
	Bit 3	0 => Línea CA2 de Entrada 1 => Línea CA2 de Salida		
CB1	Bit 4	0 => Línea CB1 activa al flanco descendente 1 => Línea CB1 activa al flanco ascendente		
CB2	Bit 5	Como en bits 1-2 y 3 pero con CB2 en lugar de CA2.		
	Bit 6	Sustituir B1, B2 y B3 por B5, B6, y B7.		
	Bit 7			

Registro Auxiliar de Control (ACR)

Determina la forma de trabajo de los puertos, de los contadores/temporizadores y del registro de desplazamiento. Responde a la dirección relativa A40BH en el Microinstructor. En la figura y tabla 8.4 aparece detallada la función de cada bit.

Memorizar entradas	Bit 0	0 ⇒ No memoriza entradas, grupo A 1 ⇒ Memoriza entradas en grupo a (Strobe en CA1)	
	Bit 1	0 ⇒ No memoriza entradas, grupo B 1 ⇒ Memoriza entradas en grupo B (Strobe en CB1)	
Registro Desplazamiento	Bit 2 y Bit 3	Entrada B4 = 0	00 ⇒ Registro # Memoria 01 ⇒ Registro desplaza con T2 10 ⇒ Registro desplaza con $\phi 2$ 11 ⇒ Registro desplaza con CB1
		Salida B4 = 1 (CB1 actúa como salida de ritmo de desplazamiento)	00 ⇒ Rotación libre con T2 01 ⇒ Desplaza con T2 10 ⇒ Desplaza con $\phi 2$ 11 ⇒ Desplaza con CB1
	Bit 4	0 ⇒ Entrada de registro desde CB2 (primero el bit 0) 1 ⇒ Salida del registro hacia CB2 (primero el bit 0)	
T2	Bit 5	0 ⇒ Línea B6 accede al contador T2 1 ⇒ Contador T2 genera interrupción solamente	
T1	Bit 6	0 ⇒ T1 cuenta una vez y luego sigue en negativo (pulso único en B7, si procede). 1 ⇒ T1 cuenta con recarga automática al llegar a cero (onda cuadrada en B7, si procede).	
	Bit 7	0 ⇒ T1 genera interrupción solamente 1 ⇒ Línea B7 actúa como salida de T1	

B7 T1	B6 T2		Registro Desplaza			Memoriza Entradas	
7	6	5	4	3	2	1	0

Fig. 8.4 Función de cada bit del registro de control

Registro de Alarmas de Interrupción (IFR)

Cada uno de sus bits advierte de la presencia de una interrupción en los diversos elementos de la VIA. En la figura 8.5 aparece detallado cada uno de sus bits.

IRQ	T1	T2	CB1	CB2	SR	CA1	CA2
7	6	5	4	3	2	1	0

Fig. 8.5 Registro de alarmas de interrupción

Indica la aparición de un estado o flanco activo en las líneas o temporizadores indicados:

0 = No se altera la alarma.

1 = Repone la alarma.

Examinando el bit 7 del registro IFR se determina si ha existido interrupción cuando se pone a 1.

Registro de Activación de Interrupciones (IER)

Cada uno de sus bits sirve para activar o desactivar las interrupciones de la VIA. Ocupa la dirección A40Eh en el Microinstructor. La figura 8.6 detalla los bits de este registro.

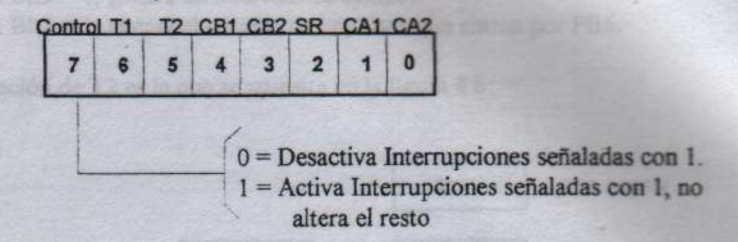


Fig. 8.6 Registro de Activación de Interrupciones.

8.1.4 Temporizadores y contadores (Timer 1)

En la VIA existen dos Timer, T1 y T2; ahora procederemos al estudio del Timer 1. En la figura 8.7 se observa su constitución y las direcciones que ocupa.

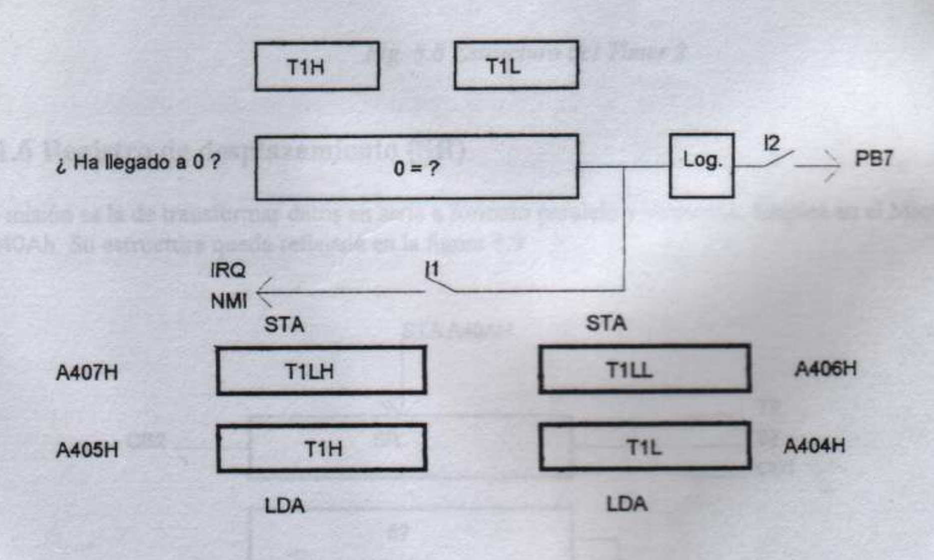


Fig. 8.7 Estructura del Timer 1

Según se desprende de las figuras para que el Timer 1 inicie el proceso de cuenta (decremento del contador), es necesario cargarlo previamente, para ello usaremos la instrucción LDA #dato, posteriormente se transfiere el contenido del acumulador a los Latch del Timer con la instrucción STA (A406H-A407H); en este instante se inicia el decremento del Timer apareciendo la señal generada en Puerto B7.

Las acciones posibles de T1 cuando su valor llega a cero, son:

- Levanta una bandera, o pone a 1 el bit señalizador IFR.
- Traspasa o no, según el interruptor I1 a la CPU la petición de interrupción.
- Afecta o no, según el interruptor I2, a una salida por la patilla PB7. Pueden ocurrir dos cosas en PB7:
  - Generar un pulso de un ciclo, al valer T1 cero.
  - Cambia el nivel del estado lógico en PB7 cada tiempo que temporiza T1.

8.1.5 Temporizador/contador (Timer 2)

Hemos visto que la función de T1 es producir temporizaciones y contajes para el mundo exterior a través de PB7, sin embargo T1 es más sencillo y solo se puede usar para generar un breve intervalo de tiempo para contar los impulso que llegan por PB6 desde el mundo exterior. El Bit 5 de ACR selecciona el uso de T2:

- ACR Bit5 = 0; genera un intervalo de tiempo.
- ACR Bit5 = 1; cuenta el número de impulsos que entran por PB6.

La configuración de T2 es la que se muestra en la figura 8.8:

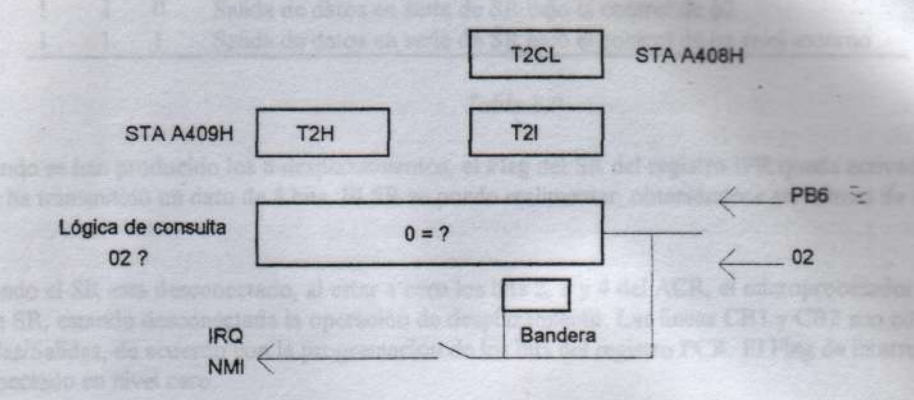


Fig. 8.8 Estructura del Timer 2

8.1.6 Registro de desplazamiento (SR)

Su misión es la de transformar datos en serie a formato paralelo y viceversa. Emplea en el Microinstructor la dirección A40Ah. Su estructura queda reflejada en la figura 8.9.

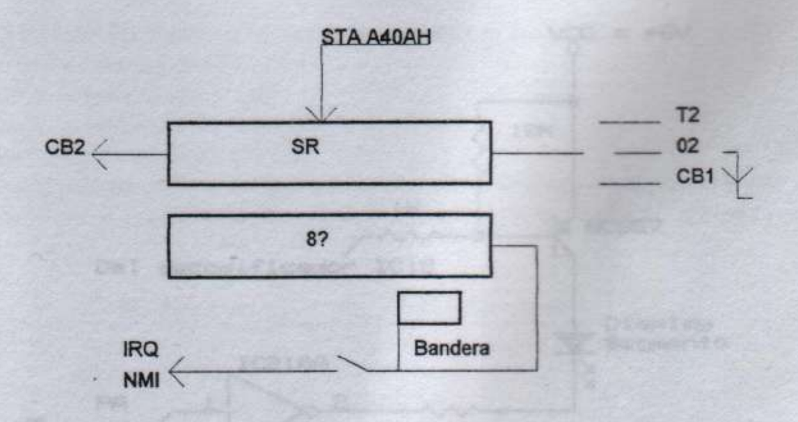


Fig. 8.9 Estructura del registro de desplazamiento

La frecuencia de los bits que controlan el desplazamiento de los bits en el registro de desplazamiento queda determinada por:

1. La activación del Flag T2.
2. $\phi 2$
3. Los flancos descendentes de CB1.

Los 8 modos de operar del registro son gobernados por los bits 2, 3 y 4 del registro (ACR), aparecen reflejados en la tabla 8.9.

Bits de ACR			
4	**3**	**2**	**Modo de funcionamiento del Registro de desplazamiento**
0	0	0	Registro de desplazamiento inhabilitado
0	0	1	Entrada de datos en serie a SR bajo el control de T2
0	1	0	Entrada de datos en serie a SR bajo el control de ϕ2
0	1	1	Entrada de datos en serie a SR bajo el control de un reloj externo
1	0	0	Salida de datos en intervalos continuos de T2
1	0	1	Salida de datos en serie de SR bajo el control de T2
1	1	0	Salida de datos en serie de SR bajo el control de ϕ2
1	1	1	Salida de datos en serie de SR bajo el control de un reloj externo

Tabla 8.9

Cuando se han producido los 8 desplazamientos, el Flag del SR del registro IFR queda activado, lo que indica que se ha transmitido un dato de 8 bits. El SR se puede realimentar, obteniéndose así formas de onda muy diversas.

Cuando el SR esta desconectado, al estar a cero los bits 2, 3 y 4 del ACR, el microprocesador puede leer o escribir sobre SR, estando desconectada la operación de desplazamiento. Las líneas CB1 y CB2 son controladas como Entradas/Salidas, de acuerdo con la programación de los bits del registro PCR. El Flag de interrupción de SR queda desconectado en nivel cero.

8.2 Entradas/Salidas locales. Teclado y display

En el Microinstructor MI-650C, el circuito encargado del control de los dispositivos locales es IC19, siendo este una VIA del tipo 6522; IC 18 es el encargado de descodificar las líneas que excitan a los displays.

El esquema más sencillo de activación del display aparece en la figura 8.10.

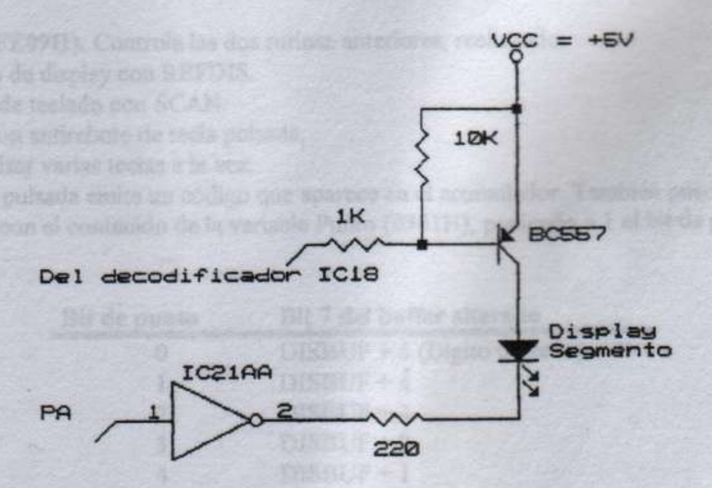


Fig. 8.10 Circuito simplificado para excitar un segmento de Display en el Microinstructor

PB0 - PB3 de IC19, controlan el decodificador IC18, el código hexadecimal en este da lugar a la activación de las 10 posibles salidas. El teclado es una matriz de 3x8, las columnas están controladas por las tres líneas de menor peso de IC18, mientras que las filas de la matriz están gobernadas por las líneas PA0- PA7 de IC19; si no se activa ninguna tecla PA0 - PA7 tendrán un nivel FFH.

La decodificación de PB0-PB3, pone a 0 la línea de salida del decodificador pasando el transistor correspondiente de corte a saturación, dando tensión al ánodo común de este. PA0-PA7 asigna los segmentos a encender según se programe. Se activará el segmento que corresponda al PA en estado alto, pues su cátodo estará a 0 V.

Para evitar el parpadeo el software realiza al menos 50 exploraciones por segundo (cada 20 ms).

8.3 Rutinas de entrada/salida

Estas son las rutinas encargadas de controlar los procesos de entrada/salida del Microinstructor, usadas por el programa monitor y que además pueden ser utilizadas por los programas de usuario.

1. REFDIS (FE4DH). Salida de datos a display. Presenta el contenido del buffer DISBUF que se encuentra en RAM (0353H) en el display durante 10 ms. El buffer esta compuesto por 6 Bytes que controlan cada uno de ellos el encendido de un dígito, correspondiendo DISBUF al dígito izquierdo y DISBUF + 5 al derecho.

Cada display se enciende cargando en DISBUF (Dirección de display a activar), el dato que activara el segmento requerido. La correspondencia de segmento y bit es la siguiente:

BIT	SEGMENTO
0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	Punto

2. SCAN (FE7DH). Analiza el teclado, detectando si hay o no una tecla pulsada. El Flag C del registro S, se pone a cero si no hay tecla pulsada y a 1 si la hay.

3. GETKEY (FE09H). Controla las dos rutinas anteriores, realizando:
- Refresco de display con REFDIS.
- Análisis de teclado con SCAN.
- Protección antirebote de tecla pulsada.
- Evita pulsar varias teclas a la vez.

Para cada tecla pulsada emite un código que aparece en el acumulador. También puede alterar el contenido de DISBUF de acuerdo con el contenido de la variable Punto (0361H), poniendo a 1 el bit de punto decimal según la tabla:

Bit de punto	Bit 7 del buffer alterado
0	DISBUF + 5 (Dígito derecho)
1	DISBUF + 4
2	DISBUF + 3
3	DISBUF + 2
4	DISBUF + 1
5	DISBUF (Dígito izquierdo)
6 y 7	Ignorado

4. OUTADD (FD22H). Coloca un valor en las 4 primeras posiciones del buffer DISBUFF.

5. OUTADH (FD2H). Coloca en DISBUFF y DISBUFF + 1 el código correspondiente al valor del Acumulador de forma similar a la anterior.

6.OUTDAT (FD32H). Coloca el código en DISBUF + 4 y DISBUF + 5 (campo de datos).

7. OUTACC (FD34H). Saca el Acumulador como dos caracteres colocándolos en DISBUF a partir de la posición indicada por rX (el máximo valor de rX será 4).

8. OUTACH (FD3DH). Saca los bits 0 a 3 del Acumulador como un carácter, colocándolo en DISBUF + X.

9. CLRDIS (FD48H). Borra el buffer del display.

10. ON (FD67H). Presenta el texto "ON" en display durante 1 segundo.

11. OFF (FD6FH). Presenta el texto "OFF" en el display durante 1 segundo.

12. GETADE (FD89H). Obtiene la dirección (4 dígitos) del teclado rotando a la izquierda hasta la pulsación de (FIN).

13. GETADD (FD90H). Igual a la anterior, pero no hay error si se pulsa (FIN) como primera tecla.

14. GETPAR (FD97H). Análoga a GETADE, pero solo obtiene dos dígitos que se colocan a la salida en "PARL".

15. GETP3 (FDB7H). Hace "PUNTO" = 30H si "MODEC" = 00H (Dato) o "PUNTO" = 3CH si "MODEC" = 01H (dirección). Luego sigue en GETP31.

16. GETP31 (FDCOH). Guarda el valor del acumulador a la entrada en "PUNTO" y pone a 1 la posición "PRIMER" para la detección de la primera pulsación en introducción de datos. Sigue en GETP32.

17. GETP32 (FDC8H). Obtiene cadenas (string) de teclas hexadecimales.

APÉNDICE A

Las direcciones de los programas de aplicación y del programa monitor del Microinstructor que se encuentran en memoria EPROM, están localizados en las siguientes direcciones:

Dirección inicial	Dirección final	Programa
E001H	E0B7H	MM-601
E0B8H	E0D0H	MM-602
E0D1H	E181H	MM603
E182H	E1CBH	MM-604
E1CCH	E4ACH	MM-605
E4ADH	E4DBH	MM-606
E4DCH	E6CAH	MM-607
E6CBH	E7D3H	MM-608
E7D4H	E8F3H	MM-615 (RMODEM)
E8F4H	EA2AH	MM-615 (TMODEM)
EA2BH	ED1BH	MM-616
ED1CH	F025H	MM-617
F026H	F0FFH	Libre
F100H	F186H	Modificaciones Monitor
F187H	F333H	Libre